МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ
ГОРОДСКОГО ХОЗЯЙСТВА имени А. Н. БЕКЕТОВА

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ, ПРАКТИЧЕСКИХ,
САМОСТОЯТЕЛЬНЫХ И КОНТРОЛЬНЫХ РАБОТ
ПО КУРСУ

ТЕХНИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

(для студентов 5-го курса дневной и 6-го курса заочной форм обучения магистров специальности 8.18010013 «Управление проектами»)

ХАРЬКОВ – ХНУГХ – 2013

Методические указания для выполнения лабораторных, практических, самостоятельных и контрольных работ по курсу «Техническое и программное обеспечение» (для студентов 5-го курса дневной и 6-го курса заочной форм обучения магистров специальности 8.18010013 «Управление проектами») / Харк. нац. ун-т гор. хоз-ва им. А. Н. Бекетова; сост.: Б. И. Погребняк, Г. В. Высоцкая. – Х.: ХНУГХ им. А. Н. Бекетова, 2013. – 120 с.

Составители: Б. И. Погребняк,
Г. В. Высоцкая

Рекомендовано кафедрой "Прикладной математики и информационных технологий", протокол № 1 от 30 августа 2012 г.

Оглавление

ВВЕДЕНИЕ

СТРУКТУРА ЛАБОРАТОРНОЙ РАБОТЫ

Каждая лабораторная работа состоит из таких частей:

- Цель работы,
- Краткие теоретические сведения,
- Ход работы,
- Индивидуальные задания и
- Контрольные вопросы.

Ход работы состоит из отдельных Шагов, которые, в свою очередь, состоят из отдельных Пунктов. В Пунктах описаны требуемые действия и получаемые (в случае правильного выполнения) результаты.

ПОРЯДОК ВЫПОЛНЕНИЯ

Лабораторные работы сконструированы так, что максимальный обучающий эффект, при минимальной затрате сил, достигается если их выполнять следующим образом:

1. Лабораторные работы необходимо выполнять **последовательно** – в том порядке, в котором они здесь приведены.
2. **Обязательно внимательно** прочитать Краткие теоретические сведения.
3. Ход работы выполнять **строго** Шаг за Шагом, и Пункт за Пунктом, **без пропусков**.
4. Имена файлов, каталогов и других объектов **вводить так**, как указано в тексте. В противном случае могут быть нарушены связи, что приведет ошибке выполнения, которая, в свою очередь, может обнаружиться через несколько Пунктов, или даже Шагов.
5. Индивидуальные задания выполнять **только после полного завершения** Хода работы.
6. Если все предыдущие пункты выполнены правильно, то ответы на Контрольные вопросы не вызовут никаких затруднений.
7. Теперь Лабораторная работа готова к защите у преподавателя, которая состоит в подтверждении полученных знаний по заданной теме.

ОБРАБОТКА ОШИБОК

Если в ходе выполнения очередного Пункта получен результат, несоответствующий требуемому, то последовательность действий должна быть такой:

1. Выполнить более внимательно текущий пункт еще раз.
2. Если результат снова не соответствует ожидаемому, то необходимо повторить весь текущий Шаг еще раз (для этого лабораторные работы и были разбиты на Шаги).
3. Если и в этом случае ошибка не была устранена, то Лабораторную работу надо начать выполнять с самого начала – с **внимательного прочтения** Кратких теоретических сведений.

СОДЕРЖАНИЕ КОНТРОЛЬНОЙ РАБОТЫ

В Контрольной работе необходимо привести выполнение Индивидуальных заданий по Лабораторным работам в соответствии со своим вариантом, и оформить их соответствующим образом. Каждая Лабораторная работа – это одна из тем освоения материала. Лабораторные работы выполняются на компьютере, а правильность их выполнения контролируется по приводимым в тексте результатам. Если полученный в ходе выполнения Лабораторной работы результат не соответствует приводимому в тексте, то необходимо вернуться на несколько шагов назад и более внимательно выполнить указанные действия. И только после правильного выполнения всех Шагов и Пунктов можно приступать к выполнению Индивидуального задания. Конкретное содержание отчета по каждой Лабораторной работе приведено в ее Индивидуальном задании.

ВЫБОР ВАРИАНТА

Вариант Индивидуального задания формируется в соответствии с:

- указанием преподавателя, или
- номером по списку группы, или
- по 2-м последним цифрам номера зачетной книжки.

По 2-м последним цифрам номера зачетной книжки вариант работы вычисляется как **остаток от целочисленного деления 2-х последних цифр номера зачетной книжки на 30** либо **последовательным вычитанием из 2-х последних цифр номера зачетной книжки по 30 до тех пор, пока остаток не будет меньше 30**. Например, две последние цифры номера зачетной книжки «07», тогда 7 % 30 = 7 или 7 mod 30 = 7, т.е. номер варианта индивидуального задания будет равен «7»; если две последние цифры номера зачетной книжки будут, например, «68», тогда номер варианта будет: 68 % 30 = 8, или 68 mod 30 = 8, или 68 – 30 = 38 – 30 = 8, т.е. «8».

ПОРЯДОК ОФОРМЛЕНИЯ

Работа оформляется на отдельных листах формата А4 (210мм × 297мм) либо в ученической тетради. Текст работы может быть набран шрифтом Times New Roman 14 пунктов через 1,5 интервала или написан от руки. На титульной странице обязательно необходимо указать название дисциплины и свой вариант.

При оформлении Индивидуальных заданиях необходимо привести:

1. Номер и название Лабораторной работы.
2. Конкретные результаты выполнения Индивидуального задания, и как они были получены (например, перечислить команды, которые использовались в работе, или HTML-коды создаваемых Web-страниц и соответствующие им эскизы, и т. д.). Более конкретное содержание отчета по каждой Лабораторной работе приведено в тексте ее Индивидуального задания.

Для дневной формы обучения необходимо также письменно ответить на Контрольные вопросы, приведенные в конце каждой Лабораторной работы, в форме «вопрос – ответ».

Настоятельно рекомендуется также в тетради для выполнения Лабораторных работ делать записи наиболее существенных (трудных, непонятных, интересных и т. п.) моментов, возникающих по ходу их выполнения.

Лабораторная работа № 1
Сеанс работы в Linux

Цель работы

Освоение и приобретение практических навыков работы с текстовым интерфейсом операционной системы (ОС) Linux.

Краткие теоретические сведения

Linux – это многозадачная, многопользовательская операционная система. Это означает, что одновременно на одном компьютере может работать много пользователей, каждый из которых одновременно может запускать на выполнение много различных программ. На современном персональном компьютере одновременно может работать только один человек. Однако многопользовательская модель имеет много преимуществ. Например, один пользователь может запустить длительный процесс (скачивание фильма из Интернета) и предать компьютер другому, или один и тот же пользователь может зарегистрироваться в системе под разными именами, и с разными полномочиями, для выполнения различных задач и т. д.

Управление работой Linux осуществляется при помощи терминала. Под терминалом (консолью) понимается устройство, предназначенное для обмена информацией между пользователем и компьютером. В минимальной конфигурации терминал состоит из монитора и клавиатуры, мышь – не обязательна. Существует два типа терминалов (интерфейсов):

1. текстовый (интерфейс командной строки – Command Line Interface – CLI) и
2. графический (Graphical User Interface – GUI).

В большинстве современных дистрибутивов Linux параметры настроены так, что по умолчанию загружается графический интерфейс (графический терминал).

ОС Linux позволяет организовать одновременную работу до двенадцати виртуальных терминалов (виртуальных консолей) – по количеству функциональных клавиш на клавиатуре. Как правило, имеется шесть текстовых виртуальных терминалов и не менее одного графического. Переключение из графического интерфейса (режима, терминала) на первый текстовый виртуальный терминал выполняется при помощи сочетания клавиш **Ctrl+Alt+F1**, на второй – **Ctrl+Alt+F2**, а на шестой – **Ctrl+Alt+F6**. Переключение виртуальных терминалов в текстовом режиме выполняется при помощи сочетаний клавиш **Alt+F$_n$**, где **n** – номер виртуального терминала. Сочетание клавиш **Alt+F7** осуществляет переключение из текстового режима в графический – седьмой виртуальный терминал. Однако, для переключения виртуальных терминалов удобнее запомнить сочетание клавиш **Ctrl+Alt+F$_n$** – оно срабатывает в обоих режимах: и в текстовом, и в графическом.

Для того, чтобы начать работу с ОС Linux пользователь должен зарегистрироваться в системе. Процедура регистрации состоит из двух шагов:

1. ввод имени
2. ввод пароля,

которые можно получить у преподавателя или системного администратора. При вводе имени и пароля необходимо обращать внимание на прописные и строчные буквы – Linux «чувствительна» к регистру символов. Пароль при вводе не отображается – чтобы его никто не смог подсмотреть. Если при вводе имени или пароля была допущена ошибка, система выдаст примерно такое сообщение: «`Login incorrect`», и процедуру регистрации придется повторить.

После успешной регистрации (начала сеанса; входа, «как в комнату»; «логинивания» и т. д.) на экране монитора появляется приглашение для ввода команды (как правило, символ «`$`»). В текстовом режиме ввод каждой команды (порции информации) завершается нажатием клавиши **`Enter`**. Команда в ОС Linux состоит из имени, за которым могут следовать ее параметры – если таковые имеются. Параметры команды от ее имени и друг от друга отделяются, по крайней мере, одним пробелом.

Каждый пользователь операционной системы Linux рано или поздно сталкивается с вопросом: «Как выполнить то, или иное действие?» При этом у новичков такие вопросы возникают более часто, у опытных пользователей – реже. Разработчики Linux снабдили ее обширной и мощной справочной системой, получить информацию в которой можно несколькими различными способами. Например, при помощи следующих команд:

- `help` (помощь),
- `man` (от. анл. «manual» – «руководство»),
- `info` (гипертекстовое руководство).

Правило, согласно которому решение любой задачи надо начинать с изучения документации, по-английски называется «Read That Fine Manual» (RTFM) – Читайте Это Прекрасное Руководство. Однако, справочная система Linux – это совсем не учебник, это – настоящий справочник. В нем содержится информация, достаточная для освоения описываемого объекта, но никаких обучающих приемов, никаких определений, повторений и выделения главного в нем обычно нет. Тем более не допускается усечение руководства с целью представить небольшие по объему, но наиболее важные сведения как это принято в учебниках. Все это преследует цель не научить, а раскрыть смысл, пояснить. Наиболее рациональный способ получения справочной информации по интересующему в данный момент объекту – открыть справку на другом виртуальном терминале.

Традиционно документация по Linux в подавляющем большинстве случаев написана на простом английском языке (даже в русифицированных дистрибутивах). Если английский – не родной язык для автора документа, он будет только проще. Так что выбор, как всегда, за Вами – учить английский, или старательно выполнять лабораторные работы…

Основной способ получения подсказки во всех UNIX-системах – команда `man`. Справочная информация в ней организована в виде так называемых страниц руководства (`manpages`). Каждая страница руководства (для краткости – просто «руководство») посвящена какому-нибудь одному объекту системы.

Страница руководства занимает, как правило, больше одного экрана. Управлять работой команды `man` просто – при помощи следующих клавиш:

- **`Q`** (от англ. «quit» – «выход из») – завершить,
- ↓ или **`Enter`** – строка вниз,
- ↑ – строка вверх,
- **`PageDn`** или **`Space`** (**`Пробел`**) – страница (экран) вниз,
- **`PageUp`** или **`B`** – страница (экран) вверх,
- **`g`** – начало текста,
- **`G`** – конец текста.
- **`?`** или **`h`** (от англ. «help» – «помощь») – исчерпывающая информация по работе с командой.

Страница руководства состоит из так называемых «полей» – стандартных разделов, с разных сторон описывающих объект. В поле `NAME` (`ИМЯ`, `НАЗВАНИЕ`) содержится краткое описание объекта – такое, чтобы его назначение было понятно с первого взгляда. В поле `SYNOPSIS` (`ОБЗОР`, `СИНТАКСИС`) дается формализованное описание способов использования объекта. В квадратные скобки в этом поле заключены необязательные параметры команды, которые можно ей передать, а можно и опустить. Текст в поле `DESCRIPTION` (`ОПИСАНИЕ`) – это развернутое описание объекта, достаточное для того, чтобы им воспользоваться. В поле `SEE ALSO` (`СМОТРИ ТАКЖЕ`) содержится список других источников по той же теме.

Все страницы руководства ОС Linux сгруппированы в восемь разделов, которые, по умолчанию, просматриваются в таком порядке:

(1) – Введение в пользовательские команды,
(8) – Введение в административные и привилегированные команды,
(2) – Введение в системные вызовы,
(3) – Введение в библиотечные функции,
(4) – Введение в специальные файлы (устройства),
(5) – Введение в форматы файлов,
(6) – Введение к играм,
(7) – Введение о соглашениях и о разном.

Команда `man` просматривает все разделы в указанно порядке и показывает первое найденное руководство с заданным именем. Чтобы посмотреть руководство по объекту из определенного раздела, необходимо в качестве первого параметра команды `man` указать номер этого раздела, например, `man 1 man`. Если в качестве первого параметра `man` использовать ключ «`-a`» то, будут последовательно выданы все руководства с заданным именем. Например, по команде «`man -a intro`» будут выданы введения (introductions) по каждому из восьми разделов справочной системы. Внутри страниц руководства принято непосредственно после имени объекта ставить в круглых скобках номер раздела, из которого выдано руководство по этому объекту. Например, `man(1)` – справочная информация по команде `man` выдана из 1-го раздела.

Команда info является некоторой альтернативой команды man. Документ info – это настоящий гипертекст, в котором множество небольших страниц объединены в древовидную структуру. В каждом разделе документа info всегда есть оглавление, из которого можно перейти сразу к нужному подразделу, откуда всегда можно вернуться обратно. Кроме того, info-документ можно читать и как непрерывный текст – поэтому в каждом подразделе есть ссылки на предыдущий и последующий подразделы. Команда info использует весь экран: на большей его части она показывает текст документа, а первая и две последних строки отведены для ориентации в его структуре.

Одна или несколько страниц, которые можно перелистывать клавишей **Пробел** (**Space**) или **PageUp**/**PageDn** – это узел (node). Узел содержит обычный текст и меню (menu) – список ссылок на другие узлы, лежащие в дереве на более низком уровне. Ссылки внутри документа имеют вид «* имя_узла::», перемещать по ним курсор можно клавишей **Tab**, а переходить к просмотру выбранного узла – клавишей **Enter**. Вернуться к предыдущему просмотренному узлу можно клавишей «**l**» (от англ. «last» – «в последний раз, далее»). И, главное, выйти из программы info можно, нажав клавишу «**q**». Более подробную справку об управлении программой info можно в любой момент получить у самой info – нажав клавишу «**?**» или «**h**». Узлы, составляющие документ info, можно просматривать и подряд, один за другим с помощью клавиш «**n**» (от англ. «next» – «потом, затем, далее») и «**p**» (от англ. «previous» – «предыдущий»), однако это бывает нужно нечасто.

В верхней строке экрана info показывает имя текущего узла, имя следующего узла и имя родительского (или верхнего) узла, в котором находится ссылка на текущий. Имя текущего узла «Top» и имя верхнего узла «dir» означают, что просматривается корневой узел документа, выше которого – только каталог со списком всех info-деревьев. В нижней части экрана расположена строка с информацией о текущем узле, а за ней – строка для ввода длинных команд (например, для поиска текста с помощью команды «/»).

Если же некоторый объект системы не имеет документации ни в формате man, ни в формате info, в этом случае – Ваш путь в каталоги /usr/share/doc/<имя_объекта>. В них содержится обширнейшая документация по системе Linux в целом, и отдельным аспектам ее применения. Большая часть этой документации представляет собой обычные текстовые ASCII-файлы, которые не имеют, увы, ни стандартного формата, ни тем более – ссылок на руководства по другим объектам системы. Для их просмотра можно воспользоваться командами more и less.

Все, с теорией закончили. Настала пора практики. В конце концов, учиться лучше всего на примерах. И дальше – по инструкции:

Ход работы

Шаг 1. Регистрация и начало работы

Чтобы начать работать в ОС Linux выполните следующее:

1. Включите компьютер (если он до этого не был включен) и дождитесь завершения загрузки.

Если Linux настроен на загрузку графического интерфейса, то в результате экран монитора должен выглядеть, примерно, так:

Если ОС Linux уже была загружена и на мониторе отображается Рабочий стол – делать ничего не надо, переходим к следующему Пункту.

2. Нажмите сочетание клавиш **`Ctrl+Alt+F2`**.

В результате управление Linux-ом будет переведено в текстовый режим. Откроется второй виртуальный терминал – черный экран и несколько строк текста, примерно, такого содержания:

```
Welcome to PSPO 5.0.4 (pspo) / tty2
computer21 login:
```

Первая строка – это приветствие Linux – дистрибутив ПСПО, а `tty2` – второй виртуальный терминал. Вторая строка содержит имя компьютера (хоста, англ. «hostname» – «хозяйское имя») – `computer21` и приглашение (`login`) ввести имя пользователя.

3. Введите имя пользователя, которое укажет преподаватель, и нажмите клавишу **`Enter`**. Например, $\texttt{student}_{\texttt{n}}$, где `n` – номер компьютера (хоста).

4. На приглашение `Password`: введите пароль, указанный преподавателем.

Например, совпадающий с именем пользователя – $\texttt{student}_{\texttt{n}}$. Обратите внимание, что при вводе пароля символы не отображаются. Если имя

пользователя и пароль введены правильно, то появится строка, примерно, такого содержания:

```
[student21@computer21 ~]$
```

В ней в квадратных скобках указано, что имя пользователя «student21», а имя хоста – «computer21». Заканчивается строка символом приглашения «$» для ввода следующей команды.

5. Перейдите на первый виртуальный терминал.

Для этого выполните команду **Ctrl+Alt+F1** (**Alt+F1**).

6. Попытайтесь зарегистрироваться под тем же именем и паролем, что и на втором виртуальном терминале, только «немножко ошибитесь» – например, имя пользователя или пароль введите с заглавной (большой) буквы.

Обратите внимание, что ОС Linux «чувствительна» к регистру символов.

7. Зарегистрируйтесь на первом виртуальном терминале под тем же именем, что и на втором.

8. Перейдите в графический терминал и зарегистрируйтесь там под тем же именем.

Для этого выполните команду **Ctrl+Alt+F7** (**Alt+F7**) и введите свое имя и пароль в соответствующие поля, в конце нажимая клавишу **Enter**.

Шаг 2. Выполнение команд

Для практического освоения приемов работы в текстовом режиме выполните следующие команды:

1. Перейдите на второй виртуальный терминал.
2. Введите команду «whoami» и нажмите клавишу **Enter**.

Команда «whoami» («Кто я?») выводит имя пользователя, который ее запустил на выполнение. Эту же информацию – имя пользователя – можно увидеть и в строке приветствия.

3. Введите команду «who».

Команда «who» («Кто») выводит имена пользователей и виртуальные терминалы («точки входа» в систему), за которыми они работают, а так же время их регистрации в системе.

4. Введите команду «who am i».

В результате будет выдана только одна строка – информация о текущем пользователе.

5. Введите команду «tty».

Будет выдано сообщение «/dev/tty2», которое говорит о том, что пользователь работает за вторым виртуальным терминалом.

6. Введите команду «make love».

Будет выдано сообщение:

«make: *** Нет правила для сборки цели `love`. Останов».

Оно говорит о том, что первое слово командной строки «make» было воспринято правильно, а со вторым – возникли проблемы. Т. е. в системе

существует команда «make», а вот ее параметр не допустим – нет такого файла в текущем каталоге.

7. Введите команду «mkae love» вместо «make love».

Сообщение:

«-bash: mkae: команда не найдена»

свидетельствует о том, что в системе команды «mkae» не существует.

8. Введите команды «Make love» и «MAKE love» вместо «make love» и объясните результат.

Шаг 3. Получение справочной информации

Для практического освоения приемов получения справочной информации в текстовом режиме выполните следующие команды:

1. Введите команду «man intro».
2. Ознакомьтесь с содержанием документа, «пролистывая» его при помощи клавиш управления курсором.
3. Для завершения просмотра документа (команды «man») – нажмите клавишу «**Q**».
4. Последовательно введите команды «help help», «help exit» и «help logout» (предварительно завершая работу с предыдущей), и ознакомьтесь с полученной информацией.
5. Выполните команду «man man».
6. Перейдите на первый виртуальный терминал.
7. Выполните команду «info man».

Сравните с информацией по команде «man» на втором виртуальном терминале.

8. Завершите работу команд «man» и «info».

Шаг 4. Завершение работы

1. Перейдите на второй виртуальный терминал.
2. Введите команду «exit».

В результате сеанс работы пользователя за вторым виртуальным терминалом будет завершен и снова появится приглашение к регистрации в системе.

3. Снова зарегистрируйтесь в системе под своим именем.
4. Введите команду «logout».
5. Перейдите на первый виртуальный терминал.
6. Введите команду «shutdown».
7. Введите команду «/sbin/shutdown».
8. Введите команду «/sbin/halt».

В текстовом режиме обыкновенный пользователь выключить компьютер не может. Выключить компьютер может только суперпользователь (администратор) – root.

9. Нажмите сочетание клавиш **Ctrl+D**.
10. Перейдите в графический режим и выключите компьютер при помощи команды **Меню ⇨ Выйти ⇨ Выключить** или **Выйти ⇨ Выключить**.

Индивидуальные задания

1. Определить сколько и какие виртуальные терминалы имеются в Вашей ОС Linux.
2. Получить справочную информацию по использовавшимся в лабораторной работе командам.
3. В тетрадь для выполнения лабораторных работ занести полученные результаты.
4. Показать работу преподавателю.

Контрольные вопросы

1. Что такое многопользовательская, многозадачная операционная система?
2. Какие преимущества предоставляет многопользовательская, многозадачная операционная система?
3. Что такое терминал (консоль), для чего он служит?
4. При помощи каких типов интерфейсов можно управлять работой ОС Linux?
5. Сколько виртуальных терминалов может быть организовано в ОС Linux?
6. Сколько текстовых виртуальных терминалов может быть в ОС Linux?
7. Сколько графических виртуальных терминалов может быть в ОС Linux?
8. Как переключиться на первый текстовый виртуальный терминал?
9. Как переключиться на первый графический виртуальный терминал?
10. Различает ли ОС Linux прописные и строчные буквы?
11. Как начать сеанс работы в Linux?
12. Что будет, если неправильно ввести имя пользователя или пароль?
13. Как (или чем) завершается ввод каждой команды (порции информации)?
14. Из каких компонентов состоит команда ОС Linux
15. Как разделяются составляющие команда Linux
16. Что делает команда «`whoami`»?
17. Что делает команда «`who`», и какие параметры она имеет?
18. Что делает команда «`tty`»?
19. Что делает команда «`make`»?
20. Какое сообщение выдает Linux, когда не может отыскать команду?
21. Какие выдаются сообщения в случае недопустимых параметров команды?
22. Какое выдается сообщение, если у пользователя не достаточно полномочий для выполнения команды?
23. Перечислите способы получения справочной информации, какая между ними разница?
24. На каком языке в большинстве случаев выводится справочная информация, и какие есть варианты?
25. Как завершить работу команд «`man`» и «`info`»?
26. Как перемещаться по тексту справки, выдаваемой командами «`man`» и «`info`»?
27. Чем отличается завершение сеанса пользователя от завершения работы ОС Linux?
28. Как завершить сеанс работы в Linux?
29. Как завершить работу Linux и выключить компьютер?
30. Можно ли завершить работу Linux и выключить компьютер из текстовой консоли?
31. Сформулируйте и задайте свой вопрос по пройденному материалу.
32. Сформулируйте и внесите свое предложение по совершенствованию темы.

Лабораторная работа № 2
Файловая система Linux

Цель работы

Освоение и приобретение практических навыков работы с файловой системой ОС Linux.

Краткие теоретические сведения

Файловая система представляет собой хранения и организации доступа к данным на внешних запоминающих устройствах (ВЗУ) компьютера (таких, например, как магнитные и оптические диски, флеш-накопители и т. д.). Любая файловая система состоит из:

1. набора правил хранения информации на ВЗУ и
2. набора программ, которые эти правила реализуют.

Каждый такой набор правил имеет свое уникальное имя – *название файловой системы*. «Родными» для операционной системы Linux являются файловые системы *Ext3fs/Ext4fs* (Extended File System ver. 3/4 – «расширенная файловая система версий 3 и 4»). Однако, она поддерживает работу и иных файловых систем, таких, например, как FAT, NTFS, CDFS и др., что обеспечивает совместимость (прямой обмен информацией) с другими операционными системами. Программы, которые реализуют работу файловых систем, входят в состав так называемой подсистемы ввода-вывода операционной системы. Более подробную информацию по файловым системам ОС Linux можно получить по команде «`man fs`».

Основными объектами файловой системы Linux являются:

1. файлы и
2. каталоги.

(В файловой системе Linux нет ни дисков, ни папок, ни документов – есть только каталоги и файлы.)

Файл – это поименованная совокупность данных на ВЗУ. В других операционных системах файлы еще иногда называют документами. Каждый файл имеет множество различных характеристик. Основной среди них, с точки зрения пользователя, являются имя файла. Имя файла, обычно, каким-то образом отражают его содержание. В Linux, в отличие от других операционных систем, не существует какого-либо формата относительно имен файлов (как, например, «8.3» в MS-DOS). В ней нет никаких предписаний по поводу расширения имени файла: в имени файла может быть любое количество точек (в том числе и ни одной), а после последней точки может быть любое количество символов. Наоборот, если имя файла начинается с символа точка «`.`» – он становится «невидимым» для некоторых команд. Имя файла в Linux может иметь длину до 255 символов и содержать любые символы, за исключением двух символов: с кодом 0 и наклонной черты (слеш) «/». Причем, Linux всегда различает прописные и строчные буквы, как в именах файлов, так и в именах каталогов.

Есть несколько символов, допустимых в именах файлов и каталогов, которые, при этом, нужно использовать с осторожностью. Это – так называемые *спецсимволы*: «~», «`», «!», «@», «#», «$», «%», «&», «*», «(», «)»,

«[», «]», «{», «}», «\», «|», «;», «:», «′», «″», «<», «>», а также символы пробела и табуляции. Если же все-таки используется один или несколько таких символов, то все имя необходимо заключить в двойные кавычки «″».

Однако, с точки зрения ОС Linux имя файла отнюдь не является его основной характеристикой. В файловой системе Linux каждому файлу однозначно соответствует так называемый «*индексный дескриптор*» (*inode*). Компьютеру удобнее работать с числами, а человек лучше воспринимает символические имена. Из этого следует, что в Linux каждый файл может иметь несколько *символических имен* (*ссылок*). Ссылки бывают двух типов:

1. жесткие (hard links) и
2. символические (symbolic links).

Жесткие ссылки – это, просто, различные имена одного и того же содержания. Количество жестких ссылок на одну и ту же область данных (файл) не ограничено, т. е. у файла может быть много разных имен. Первая жесткая ссылка (первое имя) формируется при создании файла. Последующие жесткие ссылки создаются с помощью команды «`ln`» (от англ. «link» – «соединять, связывать»), которая имеет два параметра: первый параметр – это имя файла, на который нужно создать ссылку, а второй – имя новой ссылки. После создания жесткой ссылки невозможно различить, где исходное имя, а где ссылка. Когда удаляется файл, имеющий несколько разных имен – жестких ссылок, то фактически удаляется только одна ссылка – та, которая указана в команде удаления файла. Для того, чтобы удалить файл полностью (т. е. освободить занимаемую им область, и не иметь больше возможности обращаться к этим данным) необходимо удалить все его имена – все жесткие ссылки. Номер индексного дескриптора любого файла можно узнать при помощи команды «`ls`» с ключом «`-i`». Жесткие ссылки создаются только в пределах одного носителя (одной файловой системы), их также нельзя создавать на каталоги.

Символьная (символическая) ссылка – это просто файл, в котором содержится имя другого файла. Символьные ссылки, как и жесткие, предоставляют возможность обращаться к одному и тому же файлу по разным именам. Кроме того, символьные ссылки могут указывать на файлы и каталоги в других файловых системах (т. е. на других носителях, или даже на других компьютерах), чего не позволяют жесткие ссылки. Символьные ссылки в Linux аналогичны ярлыкам в Microsoft Windows. Они называются так потому, что содержат символы – путь к файлу или каталогу. Если исходный файл или каталог удалены, символьная ссылка не удаляется, но становится бесполезной – она не будет «работать». Например, если попробовать вывести содержимое такой «битой ссылки» при помощи команд «`cat`» или «`ls`», будет выдано сообщение об ошибке. Создаются символьные ссылки при помощи той же команды, что и жесткие – «`ln`», только с ключом «`-s`» (от англ. «symbolic» – «символический»).

До тех пор, пока в системе количество файлов не слишком большое, не существует и проблемы их поиска. Если же количество файлов возрастает, например, до нескольких тысяч, то остро встает вопрос их упорядочивания и систематизации. Для решения этой задачи и были созданы (придуманы) *каталоги*.

Понятие *каталога* (*directory*) позволяет систематизировать все объекты, размещенные на носителе данных (например, на диске). В большинстве современных файловых систем используется древовидная структура организации каталогов, т.е. каталоги могут содержать другие каталоги («У больших каталогов есть маленькие каталоги…» Огастес Де Морган). Каталог, на который есть ссылка в данном каталоге, называется подкаталогом, вложенным или дочерним каталогом. Каталог, который содержит текущий каталог, называется родительским каталогом.

В любой файловой системе Linux (и UNIX вообще) всегда есть только один корневой каталог («корень», «root»), который обозначается символом «/». Пользователь Linux всегда работает с единым деревом каталогов, даже если разные данные расположены на разных носителях: нескольких жестких или сетевых дисках, съемных дисках, CD-ROM и т. п. Для того, чтобы подключать и отключать файловые системы на разных устройствах в одно общее дерево, используются процедуры монтирования и размонтирования (которые может выполнять только пользователь с правами администратора – «`root`»). После того, как файловые системы на разных носителях подключены к общему дереву, содержащиеся на них данные доступны так, как если бы все они составляли единую файловую систему: пользователь может даже не знать, на каком устройстве какие файлы хранятся.

Организация каталогов файловой системы в виде дерева не допускает появления циклов: т. е. каталог не может содержать в себе каталог, в котором содержится сам. Благодаря этому ограничению путь до любого каталога или файла всегда будет однозначным и конечным.

Имена каталогов в Linux строятся по тем же правилам, что и имена файлов. И, вообще, каталоги в принципе ничем, кроме своей внутренней структуры, не отличаются от «обычных» файлов, например, текстовых. Т. е. каждый каталог в Linux – это отдельный файл особого типа («`d`» от англ. «directory»), отличающийся от обычного файла с данными тем, что в нем могут содержаться только ссылки на другие файлы и каталоги.

В каждый момент времени пользователь работает только с одним, так называемым, «*текущим*» каталогом. Текущий каталог – это каталог, в котором запущенные программы пользователя по умолчанию (т. е. если явно не указан другой каталог) читают и записывают данные. В Linux имеется специальная команда «`pwd`» (аббревиатура от англ. «print working directory» – «напечатать (вывести) рабочий каталог»), которая возвращает имя (полный путь) текущего каталога.

Путем (англ. «path, pathname»; «тропой» к файлу) называется список каталогов, который необходимо «пройти в дереве каталогов» (маршрут, траектория), чтобы попасть в целевой каталог. Каталоги в пути друг от друга отделяются тем же символом «/», который служит для обозначения корневого каталога. Поэтому символом «/» нельзя использовать в менах файлов и каталогов. Различают полный путь, который начинается в корневом каталоге (и начинается символом «/»), и относительный путь, который начинается в текущем каталоге (и, соответственно, не начинается символом «/»).

Кроме текущего каталога в ОС Linux для каждого пользователя определен еще его «*домашний каталог*» (аналог папки «*Мои документы*» Microsoft Windows) – каталог, предназначенный для хранения собственных данных пользователя. В этом каталоге пользователь имеет все права: может создавать и удалять файлы и каталоги, менять права доступа к ним и т. д. В каталоговой структуре Linux домашние каталоги пользователей обычно размещаются в каталоге `/home` и имеют имена, совпадающие с именами пользователей. Например, `/home/student21` – полный путь в домашний каталог пользователя `student21`. Каждый пользователь может обращаться к своему домашнему каталогу с помощью символа «тильда» – «~» (в системе не существует каталога с именем «~» – это просто «синтаксический сахар»). Поэтому строка «`~/temp`» означает каталог `temp`, который находится в домашнем каталоге. Когда пользователь входит в систему, текущим каталогом становится его домашний каталог.

Для смены текущего каталога (перемещения, «путешествия» по файловой системе) служит команда «`cd`» (от англ. «change directory» – «сменить каталог»). Команда «`cd`» принимает один параметр – имя каталога (полный или относительный путь), в который нужно переместиться, т. е. сделать текущим. Команда «`cd`» без параметров эквивалента команде «`cd ~`» и делает текущим каталогом домашний каталог пользователя.

Команда «`ls`» (от англ. «list» – «список») без параметров служит для вывода на экран содержимого текущего каталога – имен файлов и вложенных подкаталогов. Если нужно просмотреть содержимое не текущего, а какого-то другого, надо явно указать команде «`ls`» полный или относительный путь к этому каталогу. Для получения дополнительной информации о файлах и каталогах используются дополнительные параметры – *ключи* (*опции*, *флаги*). Например, команда «`ls`» с ключом «`-l`» (от англ. «long» – «длинный») выводит расширенную информацию о содержании каталога. Помимо имен файлов и каталогов в этом случае отображается информация об их владельцах, количестве жестких ссылок, правах доступа, размере и дате последней модификации. Если задать ключ «`-i`» (от англ. «index»), то будут отображены номера индексных дескрипторов. Ключ «`-F`» к каждому имени добавляет суффикс: символ «`/`» – для каталогов, символ «`*`» – для программ (исполнительных, двоичных файлов). Для отображения скрытых файлов (имена которых начинаются на символ точка «`.`») служит ключ «`-a`» (от англ. «all» – «все»). Ключ «`-d`» (directory) служит для того, чтобы команда «`ls`» выводила информацию не о содержимом каталога, а о самом этом каталоге (как о файле, который содержит список ссылок на файлы и вложенные каталоги). Порядок следования ключей в команде произвольный; их так же можно перечислять как отдельно, так и объединять, например, «`ls -l -i`», «`ls -I -l`», «`ls -li`», или «`ls -il`».

На структуру каталогов UNIX-подобных операционных систем существует довольно строгий стандарт – так называемый Filesystem Hierarchy Standard (FHS). Он регламентирует не только расположение основных каталогов, но и их

подкаталоги, а иногда даже приводит список конкретных файлов, которые должны присутствовать в определенных каталогах. Этот стандарт последовательно соблюдается во всех Linux-системах. Корневой каталог большинства ОС Linux имеет, примерно, такое содержание:

- `/bin` (от англ. «binaries» – «двоичные, исполняемые») – в этом каталоге находятся исполняемые файлы самых необходимых утилит (команд) Linux, таких, например, как «`pwd`», «`ls`», «`mv`» и т. п.
- `/boot` («загрузка системы») – в этом каталоге находятся файлы, необходимые для загрузки системы, в частности само ядро Linux.
- `/dev` (от англ. «devices» – «устройства») – в этом каталоге находятся все имеющиеся в системе драйверы устройств – они используются для доступа к устройствам и ресурсам системы, таким, например, как диски, модемы, память и т. д.
- `/etc` (от англ. «et cetera» – «разное, и так далее») – содержит различные служебные файлы, в частности, – системные конфигурационные файлы.
- `/home` («домой») – здесь расположены домашние каталоги пользователей.
- `/lib` (от англ. «libraries» – «библиотеки») – в этом каталоге находятся образы разделяемых библиотек (shared library images) – файлы, содержащие код, который могут использовать многие программы.
- `/lost+found` («потерять и найти») – этот каталог используется при восстановлении файлов системы командой `fsck`.
- `/mnt` (от англ. «mount» – «монтировать») – каталог для монтирования – временного подключения файловых систем, например, на съемных носителях (CD-ROM и др.).
- `/proc` – это «виртуальная файловая система», в которой файлы хранятся в памяти, а не на диске; они связаны с различными процессами, происходящими в системе, и позволяют получить информацию о том, что делают программы и процессы в указанное время.
- `/root` – домашний каталог администратора системы – пользователя «`root`».
- `/sbin` (от англ. «system binaries») – в дополнение к утилитам `/bin` здесь находятся программы, необходимые для загрузки, резервного копирования и восстановления системы, полномочия, на исполнение которых, есть только у системного администратора.
- `/tmp` (от англ. «temporaries» – «временные файлы») – этот каталог предназначен для временных файлов: в таких файлах программы хранят промежуточные данные, необходимые им только на время работы; после завершения работы программы временные файлы теряют смысл и, как правило, удаляются.
- `/usr` – это «государство в государстве». Здесь можно найти такие же подкаталоги `bin`, `etc`, `lib`, `sbin`, как и в корневом каталоге. Однако в корневой каталог попадают только утилиты, необходимые для загрузки и восстановления системы в аварийной ситуации, все остальные программы и данные располагаются в подкаталогах `/usr`.
- `/var` (от англ. «variable» – «переменные») – здесь размещаются те данные, которые создаются в процессе работы разными программами и

предназначены для передачи другим программам и системам (очереди печати, электронной почты и др.) или для сведения системного администратора (системные журналы, содержащие протоколы работы системы). В отличие от каталога `/tmp` сюда попадают те данные, которые могут понадобиться после того, как создавшая их программа завершила работу.

Более подробно с основными каталогами Linux можно познакомиться по тексту стандарта FHS, полностью который можно найти по адресу http://www.pathname.com/fhs/.

Файловая система ОС Linux поддерживает следующие типы файлов, которые обозначаются первым символом в поле прав доступа, и принимающего такие значения:

- «`-`» – обычный файл – может содержать текстовые данные, музыку, фильм, программу и т. п.
- «`d`» – каталог – файл, который может содержать только список ссылок на другие вложенные файлы и каталоги.
- «`b`» – файл блочного устройства – устройство, запись и чтение информации в котором выполняется блоками, например, жесткий диск.
- «`c`» – файл символьного устройства – устройство, запись и чтение информации в котором выполняется посимвольно, например, терминал.
- «`s`» – доменное гнездо (от англ. «socket») – это соединение между процессами, которое позволяет им взаимодействовать, не подвергаясь влиянию других процессов. Они являются ключевым понятием TCP/IP и, соответственно, на них целиком строится Интернет. Среди стандартных средств, использующих гнезда – система *X Window*, система печати и система *syslog*.
- «`p`» – именованный канал (от англ. «pipe» – «труба») – или буфер FIFO (First In – First Out) служит в основном для того, чтобы организовать обмен данными между разными приложениями (процессами). Все, что один процесс помещает в канал, другой может оттуда прочитать. Именованные каналы создаются при помощи команды «`mkfifo`».
- «`l`» – символическая ссылка (от англ. «link») – файл, который может содержать только символический указатель (ссылку) на другой файл.

Определить тип файла можно и на основании его содержания. Так, многие форматы файлов предусматривают указание в начале файла, как следует интерпретировать дальнейшую информацию: как программу, исходные данные для текстового редактора, страницу HTML, звуковой файл, изображение или что-то другое. Для этого в Linux есть команда «`file`», которая читает первые несколько сотен байт файла, ищет в них ключевые последовательности символов и на этом основании делает предположение о типе файла.

В Linux для отделения пути к файлу от его имени служат команды «`dirname`» и «`basebane`», соответственно. Например, команда «`basename /etc/fstab`» выдаст имя файла «`fstab`», а команда «`dirname /etc/fstab`» – путь к нему – «`/etc`». Однако, команды «`dirname`» и «`basebane`» никак не анализируют тип файла – они просто

разделяют последнюю компоненту полного имени и все остальное. Например, команды «dirname /etc/foomatic» и «basename /etc/foomatic» выдадут «/etc» и «foomatic», соответственно, – они просто отделили последнюю компоненту «foomatic». Хотя «foomatic» – это каталог («ls -ldF /etc/foomatic»).

Поскольку Linux – система многопользовательская и многозадачная, вопрос разграничения прав доступа к файлам и каталогам является одним из существенных моментов функционирования операционной системы (необходимо защитить файлы каждого пользователя от «дурного влияния» других пользователей).

В основе механизма разграничения доступа лежат имена пользователей и имена групп пользователей. В Linux каждый пользователь имеет уникальное имя, под которым он входит в систему. Кроме того, в системе создается некоторое число групп пользователей, причем каждый пользователь может быть включен в одну или несколько групп. Члены разных групп могут иметь разные права по доступу к файлам и каталогам. Например, группа администраторов имеет больше прав, чем группа программистов. Создает и удаляет группы пользователей суперпользователь («root»), он же может изменять состав той или иной группы.

Например, команд «ls -l /bin/ls» отображает информацию о самой себе – о бинарном (исполнительном, программном) файле «/bin/ls», который запускается по команде «ls»:

```
-rwxr-xr-x 1 root root 92136 Сен 20 2009 /bin/ls
```

Где, справа налево: «/bin/ls» – полное имя файла, «Сен 20 2009» – дата последней модификации файла, «92136» – размер файла (в байтах), «root» – имя группы, «root» – имя владельца файла, «1» – количество жестких ссылок на файл, «rwxr-xr-x» – права доступа к файлу, «-» – тип файла (обычный файл).

Права доступа (второе поле слева) подразделяются на три типа: чтение (read), запись (write) и выполнение (execute). Эти типы прав доступа могут быть предоставлены трем классам пользователей: владельцу файла, группе, в которую входит владелец, и всем прочим пользователям.

Разрешение на чтение позволяет пользователю читать содержимое файлов, а в случае каталогов – просматривать перечень имен файлов в каталоге (например, по команде «ls»). Разрешение на запись позволяет пользователю писать в файл и изменять его. Для каталогов это дает право создавать и удалять в нем вложенные файлы и каталоги. Наконец, разрешение на выполнение позволяет пользователю выполнять файлы (как бинарные программы, так и командные файлы). Разрешение на выполнение применительно к каталогам означает: во-первых, сделать этот каталог текущим (команда «cd»), а во-вторых, обращаться за доступом к содержащемся в нем файлам. Тогда для файла /bin/ls: владелец («root») имеет все права (rwx – читать, изменять и выполнять), а группа и все остальные пользователи – только читать и выполнять, но не изменять (r-x).

Важно заметить, что права доступа, которые имеет файл, зависят также от прав доступа к каталогу, в котором этот файл находится. Например, даже если файл имеет «-rwxrwxrwx», другие пользователи не смогут до него добраться, если у них не будет прав на чтение и выполнение каталога, в котором находится файл. Например, если пользователь захочет ограничить доступ ко всем своим файлам, он может просто изменить права доступа своего домашнего каталога на «drwx------». Таким образом, никто другой не будет иметь доступ в его каталог (кроме «root»), а, следовательно, посторонним будут недоступны и все файлы – так что можно не заботиться об индивидуальной защите своих файлов. Другими словами, чтобы иметь доступ к файлу, необходимо иметь доступ ко всем каталогам, лежащим на пути от корня к этому файлу, а также разрешение на доступ собственно к этому файлу. При обращении к файлу или каталогу, доступа к которому у пользователя нет, ему будет выдано сообщение вроде «Permission denied» – «Отказано в доступе».

В символической ссылке не используются права доступа – они всегда отображаются, как «lrwxrwxrwx». Вместо этого, права доступа к файлу, полученному символической ссылкой, определяются правами доступа к файлу, на который она ссылается.

Для установки (изменения) прав доступа к файлу служит команда «chmod» (от англ. «change mode» – «сменить режим (доступа)»), которая имеет такой синтаксис:

```
chmod {a|u|g|o}{+|-}{r|w|x} <имя файла>,
```

где:

- применить к:
  - «a» (от англ. «all» – «все») – пользователям всех категорий (владельцу, группе и всем остальным),
  - «u» (от англ. «user» – «пользователь») – владельцу файла,
  - «g» (от англ. «group» – «группа») – группе или
  - «o» (от англ. «other» – «другие») – всем остальным пользователям;
- права:
  - «+» – добавить,
  - «-» – лишить;
- на:
  - «r» (от англ. «read») – чтение,
  - «w» (от англ. «write») – запись,
  - «x» (от англ. «execute») – выполнение (использование).

Например, команда:

```
chmod a+x file_name
```

предоставляет всем пользователям системы право на выполнение файла file_name, команда:

```
chmod go-rw file_name
```

лишает права на чтение и запись для всех, кроме владельца файла, а команда:

```
chmod ugo+rwx file_name
```

дает всем право на чтение запись и выполнение. Если опустить указание на то, кому предоставляется данное право, то подразумевается, что речь идет вообще обо всех пользователях, т. е. вместо:

```
chmod a+x file_name
```

можно записать просто:

```
chmod +x file_name.
```

Выполнять смену права доступа к файлу или каталогу с помощью команды «chmod» может только владелец или суперпользователь («root»). Для того, чтобы иметь возможность изменять права группы, владелец должен дополнительно быть членом той группы, которой он хочет дать права на данный файл. Изменять владельца и группу, возможно при помощи команд «chown» и «chgrp», соответственно, выполнять которые может только суперпользователь.

Для создания каталогов в ОС Linux служи команда «mkdir» (от англ. «make directory» – «создать каталог»), которая в качестве единственного обязательного параметра принимает имя (путь) создаваемого каталога. Например, команда «mkdir my_dir» создает в текущем каталоге подкаталог с именем my_dir. Если теперь выполнить команду «ls –a my_dir», то она выдаст список из двух имен «.» и «..» – во вновь созданном каталоге автоматически создаются две записи: «.» – ссылка на этот каталог и «..» – ссылка на родительский каталог. Хотя создавать жесткие ссылки на каталог нельзя, команда «ls –ld my_dir», помимо всего прочего, покажет, что тип файла «d» – каталог, который имеет две жесткие ссылки: собственное имя – «my_dir» и имя «.». Если же в каталоге создать подкаталог, количество жестких ссылок на этот каталог увеличится на одну за счет имени «..» в созданном подкаталоге.

Для удаления ненужных объектов файловой системы – файлов и каталогов – служат команды «rm» (от англ. «remove» – «удалять») и «rmdir» (от англ. «remove directory» – «удалить каталог»), соответственно. Команда «rm» предназначена именно для удаления жестких ссылок, а не самих файлов. В Linux, чтобы полностью удалить файл, требуется последовательно удалить все жесткие ссылки на него. При этом все жесткие ссылки на файл (его имена) равноправны – среди них нет «главной», с исчезновением которой исчезнет файл. Пока есть хоть одна ссылка, файл (его содержание) продолжает существовать. Для того, чтобы воспользоваться командами «rm» и «rmdir», необходимо иметь право на запись в каталог, в котором расположены удаляемые файлы или каталоги. При этом полномочия на изменения самих файлов не обязательны. Для удаления каталога необходимо в начале удалить в нем все файлы и вложенные подкаталоги, а только затем удалить пустой

каталог при помощи команды «rmdir». Однако, можно удалить и непустой каталог, если воспользоваться командой «rm» с ключом «-r» (от англ. «recursive» – «рекурсивно»). Команда «rm -r <имя каталога>» – очень удобный способ потерять в одночасье все файлы: она рекурсивно обходит весь каталог, удаляя все, что попадется: файлы, подкаталоги, символьные ссылки. При этом всегда необходимо помнить, что в Linux не предусмотрено процедуры восстановления удаленных файлов и каталогов (Удаленные файлы и каталоги сразу попадают в печь, а не в мусорную корзину, и восстановить их из пепла уже не удастся!). Большинство дистрибутивов Linux настроены так, что удаление файла или каталога происходит «молча» – никакого предупреждающего сообщения при этом не выдается. Для выдачи дополнительного запроса на подтверждение операции удаления служит ключ «-i», пренебрегать которым не стоит.

В Linux для копирования файлов и каталогов предназначена утилита «cp» (от англ. «copy» – «копировать»), которая требует присутствия двух обязательных параметров: первый – копируемый файл или каталог (откуда), второй – файл или каталог назначения (куда). При этом нужно иметь в виду, что при копировании файла или каталога поверх уже существующего не выводится никакого предупреждения! (Говоря о копировании, уместно вспомнить широко известное высказывание, приписываемое Уильяму Оккаму: «Не следует умножать сущности сверх необходимого».) Впрочем, можно использовать команду «cp» с ключом «-i», тогда перед записью поверх существующего файла будет запрашиваться подтверждение.

Если необходимо не скопировать, а переместить файл из одного каталога в другой – для этого существует команда «mv» (от англ. «move» – «перемещать»). Синтаксис и опции у нее такие же, как и команды «cp». Она сначала копирует файл или каталог, а только потом удаляет исходный. Команда «mv» может использоваться не только для перемещения, но и для переименования файлов и каталогов – для этого необходимо просто задать в качестве аргументов старое и новое имя. При использовании команды «mv», также как и при использовании команды «cp», рекомендуется использовать опцию «-i» для того, чтобы получить предупреждение, когда файл будет перезаписываться.

Многоцелевая команда «cat» (от англ. «concatenate» – «объединить»), служит для:

- конкатенации (объединения, слияния) нескольких файлов в один. Например, команда:

```
cat file1 file2 file3 > new_file
```

объединяет три файла в один, т. е, file1 + file2 + file3 = new_file.

- копирования файлов. Например, команда:

```
cat file1 > file2
```

копирует file1 в file2.

- создания текстовых файлов. Например, команда:

```
cat > new_file
```

создает файл `new_file` на основе ввода со стандартного устройства ввода – клавиатуры. Для завершения ввода (и команды «`cat`») служат сочетания клавиш **`Ctrl+D`** и **`Ctrl+C`**. Сочетание клавиш **`Ctrl+C`** посылает программе, которая считывает с клавиатуры, сигнал аварийного прекращения работы, а **`Ctrl+D`** сообщает ей, что закончен ввод очередной строки, и можно продолжать работу дальше; если же строка пустая – то закончить работу программы. Можно считать, что **`Ctrl+C`** – это сокращение от английского «Cancel» («Отмена»), а **`Ctrl+D`** – от «Done» («Сделано»).

- отображения содержимого файла. Например, команда:

```
cat file
```

выводит содержимое указанного файла `file` на системное устройство вывода – на дисплей. Однако, команду «`cat`» удобно использовать для вывода только очень небольших по объему файлов – содержимое большого файла мгновенно проскакивает по экрану, и пользователь видит только последние строчки.

Для просмотра больших по объему файлов более удобно пользоваться командами «`more`» («более») и «`less`» («менее»). Команда «`more`» выводит содержимое файла постранично. Для того, чтобы увидеть следующую страницу, надо нажать клавишу пробел, а предыдущую – «**`B`**». Нажатие клавиши **`Enter`** приводит к смещению на одну строку. Для завершения работы команды (не пролистывая весь файл до конца) необходимо нажать клавишу «**`Q`**». Команда «`less`» содержит все функции команды «`more`» и некоторые дополнительные. Управлять работой команды «`less`» просто – при помощи следующих клавиш:

- **`Q`** (от англ. Quit) – завершить,
- ↓ или **`Enter`** – строка вниз,
- ↑ – строка вверх,
- **`PgDn`** или **`Space`** (**`Пробел`**) – страница (экран) вниз,
- **`PgUp`** или **`B`** – страница (экран) вверх,
- **`g`** – начало текста,
- **`G`** – конец текста.
- **`?`** или **`h`** (от англ. «help» – «помощь») – исчерпывающая информация по работе с командой.

Кстати, для отображения страниц интерактивного руководства команда «`man`» по умолчанию использует команду «`less`».

Для поиска файлов и каталогов в ОС Linux используется адаптируемая команда «`find`» («поиск»). Она может искать файлы по имени, размеру, дате создания или модификации и некоторым другим критериям, а также выполнять различные операции над найденными объектами.

Общий синтаксис команды «`find`» имеет следующий вид:

```
find [<список каталогов>] <критерий поиска>.
```

Параметр <список каталогов> определяет область поиска. Если задать в качестве начального каталога поиска корневой каталог «/», то поиск может затянуться очень надолго, т. к. будет просматриваться вся структура каталогов, включая смонтированные файловые системы (в том числе и сетевые, если таковые имеются). Если не указать ни одного пути, поиск будет производиться только в текущем каталоге и его подкаталогах.

Началом <критерия поиска> считается первый аргумент, начинающийся на один из символов: «-», «(», «)», «,», «!». Все аргументы, предшествующие <критерию поиска>, трактуются как имена каталогов, в которых необходимо производить поиск.

Чаще всего поиск производится по имени файла. Например, команда:

```
find /usr/share/doc /usr/doc /usr/local/doc -name
Mousepad
```

в указанных каталогах ищет расположение документации по текстовому редактору *Mousepad*, команда:

```
find /usr/share/doc /usr/doc /usr/local/doc -name xfce*
```

ищет документацию по графическому интерфейсу *xfce*, а команда:

```
find -name file_name?
```

в текущем каталоге ищет файлы с именами file_name1, file_name2, file_namec и т. п.

Для поиска файлов и каталогов, имена которых известны не совсем точно, наиболее часто используются следующие символы подстановки (шаблона, замены):

- «*» – означает любое количество любых символов и
- «?» – один любой символ.

Например, по шаблону «*.doc» будут найдены все файлы с любым именем и расширением «.doc», по шаблону «file_name.*» – файлы с именем «file_name» и любым расширением. Символы подстановки используются не только в команде «find», но и во многих других командах Linux для массовой обработки файлов одной командой. Более детальную информацию по команде «find» можно получить на интерактивных man-страницах.

При редактировании и сохранении файлов довольно часто возникает зада сравнения различных их версий (копий), а также автоматического внесения изменений в более старую версию. Для сравнения файлов в ОС Linux имеется команда «diff» («разность»), которая сравнивает два файла и выдает информацию об имеющихся различиях. Если эти различия сохранить в отдельном файле, то затем при помощи команды «patch» («лоскут», «заплатка») их можно внести в более старую версию файла. Например, команда:

```
diff file1 file2 > file.diff
```

записывает в файл file.diff отличия между файлами file1 и file2, а команда:

```
patch file1 file.diff
```

применяет к файлу file1 «заплатки» – файл file.diff. В результате содержимое файла file1 становится таким же, как и file2. Такая процедура используется, например, если над одним файлом работает несколько человек одновременно в разных местах и рассылка только изменений позволяет каждому автору самостоятельно решать, применять их или нет.

Для работы с объектами файловой системы в ОС Linux наиболее часто используются следующие команды:

- pwd – отобразить полный путь в текущий каталог.
- ls [<ключи>] [<имя файла или каталога>] – отобразить содержимое каталога или параметры файла.
- file [<ключи>] [<имя файла>] – определить тип файла.
- dirname <полное имя файла> – вывести имя каталога из заданного полного имени.
- basebane <полное имя файла> – вывести имя файла из заданного полного имени.
- cd [<имя каталога (путь)>] – сменить текущий каталог.
- mkdir [<ключи>] <имя каталога> – создать каталог.
- rm [<ключи>] <имя файла или каталога> – удалить файл или каталог.
- rmdir [<ключи>] <имя каталога> – удалить непустой каталог.
- cp [<ключи>] <имя исходного файла или каталога (откуда)> <имя результирующего файла или каталога (куда)> – копировать файлы и каталоги.
- mv [<ключи>] <имя исходного файла или каталога (откуда)> <имя результирующего файла или каталога (куда)> – переместить или переименовать файлы и каталоги.
- cat [<ключи>] <имена файлов> – объединить файлы или копировать, в том числе, на стандартный вывод – монитор.
- more [<ключи>] <имя файла> – постранично отобразить содержимое файла на стандартном устройстве вывода – мониторе.
- less [<ключи>] <имя файла> – постранично отобразить содержимое файла на стандартном устройстве вывода – мониторе.
- ln <существующее имя файла> <новое имя> – создать жесткую ссылку.
- ln –s <существующее имя файла или каталога> <новое имя> – создать символическую ссылку.
- find [<список каталогов>] <критерий поиска> – найти файл или каталог.
- diff [<ключи>] <имя файла1> <имя файла2> – найти различия между двумя файлами.
- patch [<ключи>] [<имя исходного файла> [<заплатки>]] – применить «заплатки» к исходному файлу.
- chown <новое имя владельца> <имя файла> – сменить владельца файла (может выполнять только суперпользователь).

• chgrp <новое имя группы> <имя файла> – сменить группу (может выполнять только суперпользователь или владелец файла, но он должен входить в группу, которой должен дать права на файл).

Все, с теорией закончили. Настала пора практики. В конце концов, учиться лучше всего на примерах. И дальше – по инструкции:

Ход работы

Шаг 1. Исследование системных каталогов

Для практического ознакомления с содержанием системных каталогов выполните следующее:

1. Введите команду «ls -F /».

Будет выведено содержание корневого каталога ОС Linux на Вашем компьютере. Наклонная черта «/» в конце команды говорит о том, что это каталог.

2. Введите команду «ls -F /bin».

Вы можете обнаружить здесь уже знакомые Вам команды, вроде «cp», «ls» и «mv». Это и есть программы соответствующих этим команд. Когда, например, пользователь вводит команду «ls», то выполняется программа /bin/ls. Символ «*» в конце имени говорит о том, что это программа (двоичный, исполняемый файл), символ «@» – что это символическая ссылка, а отсутствие какого-либо символа – что это обычный файл.

3. Введите команду «ls -F /etc».

Будет выведено содержание каталога настроек ОС Linux.

4. Введите команду «file /etc/*».

Посмотрите информацию о типе файлов, содержащихся в каталоге /etc.

5. Введите команду «ls /etc».

Будет выведен список имен, содержащихся в каталоге /etc, на основании которого трудно определить, что стоит за каждым из них.

6. Введите команду «basename /etc/fstab».

Из полного имени будет выделена последняя компонента – имя файла fstab.

7. Введите команду «dirname /etc/fstab».

Из полного имени будет выделен путь (без имени файла) – /etc.

8. Введите команду «ls -ldF /etc/foomatic».

9. Выполните команды basename и dirname для каталога «/etc/foomatic».

Объясните результат.

10. Посмотрите содержание других общесистемных каталогов.

Для некоторых каталогов может быть выдано сообщение вроде «Permission denied» – «Отказано в доступе» – значит у Вас нет прав доступа к этому каталогу.

Шаг 2. Исследование домашнего каталога

Для исследования домашнего каталога выполните следующее:

1. Введите команду «cd».

Если до этого времени текущим каталогом был какой-либо другой каталог, команда «cd» без параметров выполнит переход в домашний каталог пользователя.

2. Введите команду «pwd».

В результате будет выдана информация о домашнем каталоге пользователя, например:

```
/home/student21
```

Имя домашнего каталога по умолчанию совпадает с именем пользователя, а все домашние каталоги пользователей находятся в каталоге первого уровня /home.

3. Введите команду «ls».

Ознакомьтесь с содержанием своего домашнего каталога.

4. Введите команду «ls -a».

Оказывается, в домашнем каталоге есть масса скрытых файлов – имена которых начинаются на символ точка «.». Эти файлы служат, в основном, для настройки «рабочей среды» пользователя. Без особой надобности их изменять не следует – в лучшем случае изменится привычный «внешний вид» системы, в худшем случае – система может перестать работать вообще.

5. Посмотрите, домашние каталоги каких пользователей еще имеются в системе.

6. Попытайтесь посмотреть содержимое домашних каталогов других пользователей.

Шаг 3. Создание и перемещение по каталогам

Для практического освоения приемов создания и смены текущего каталога выполните следующее:

1. Введите команду «cd».

Чтобы перейти в свой домашний каталог (если текущим был не он), и по умолчанию создавать в нем файлы и другие каталоги (подкаталоги).

2. Выполните команду «mkdir <Ваша Фамилия>». Например, «mkdir Иванов».

В результате в домашнем каталоге будет создан каталог (подкаталог, вложенный каталог) <Ваша Фамилия>.

3. Выполните команду «cd <Ваша Фамилия>». Например, «cd Иванов».

В результате текущим каталогом станет каталог <Ваша Фамилия>, что сразу же отобразится в строке приветствия и она примет, примерно, такое содержание:

```
[student21@computer21 Иванов]$
```

4. Выполните команду «ls -a», чтобы посмотреть содержимое вновь созданного пустого каталога <Ваша Фамилия>.

Будет отображено две ссылки: «.» – ссылка на вновь созданный каталог <Ваша Фамилия> и «..» – ссылка на родительский каталог – домашний каталог пользователя «~».

5. Создайте каталог <Ваше Имя>.

6. Перейдите в каталог <Ваше Имя>.

Строка приветствия при этом изменится и примет, например, такой вид:

```
[student21@computer21 Иван]$
```

7. Посмотрите содержание текущего каталога.

8. Выполните команду «`cd ..`».

В результате Вы переместитесь «на один этаж вверх», т. е. в родительский каталог – `<Ваша Фамилия>`, что отобразится в строке приветствия.

9. Выполните команду «`cd .`».

Посмотрите на строку приветствия. Почему она не изменилась?

10. Выполните команду «`pwd`».

Команда «`cd .`» текущего каталога не меняет. «`.`» – это ссылка на текущий каталог (на самого себя).

Шаг 4. Создание и просмотр текстовых файлов

Для практического освоения приемов создания и просмотра текстовых файлов выполните следующее:

1. Введите команду «`cat`».

Курсор переместится в начало новой строки – система ждет ввода от пользователя.

2. Наберите текст «Меня зовут (например, Иван)» и нажмите клавишу **`Enter`**.

Введенный текст отобразится с новой строки, курсор переместится на следующую строку – система вновь ждет ввода от пользователя.

3. Нажмите сочетание клавиш **`Ctrl+D`**.

Команда «`cat`» завершит работу и снова появится строка приглашения для ввода команды.

4. Еще раз введите команду «`cat`».

5. Наберите произвольный текст и нажмите сочетание клавиш **`Ctrl+D`** два раза.

Первое нажатие клавиш **`Ctrl+D`** вызовет отображение введенного текста в той же строка, а второе – завершение команды `cat`.

6. Снова ведите команду «`cat`».

7. Наберите произвольный текст и нажмите сочетание клавиш **`Ctrl+C`**.

В конце набранного текста появятся символы «`^C`» – визуальный эквивалент сочетания клавиш **`Ctrl+C`**, и команда «`cat`» завершит работу.

8. Введите команду «`cat > <Ваше Имя>1.txt`». Например, «`cat > Иван1.txt`».

Символ «>» в командной строке – это операция перенаправления вывода – приводит к тому, что все, что вывелось бы на экран терминала, попадает в файл. Если файл с таким именем уже существует, то он будет перезаписан, – старое содержание уничтожится. Для того, чтобы старое содержание файла не уничтожать, а новые данные добавить в конец, существует другая операция – добавление – «>>». Более подробно работа с этими, и аналогичными возможностями, рассмотрена в теме *Shell*.

9. Введите текст «Меня зовут (например, Иван)» и нажмите клавишу **`Enter`**.

10. Для завершения ввода нажмите сочетания клавиш **`Ctrl+D`** или **`Ctrl+C`**.

11. Введите команду «`cat <Ваше Имя>1.txt`». Например, «`cat Иван1.txt`».

Будет выведено содержание файла <Ваше Имя>1.txt: «Меня зовут …».

12. Создайте файл <Ваше Имя>3.txt и запишите в него строку «Сегодня: ».

13. Просмотрите содержимое файла <Ваше Имя>3.txt.

14. Выполните команду «date >> <Ваше Имя>3.txt».

15. Просмотрите содержимое файла <Ваше Имя>3.txt.

Будет выведено содержание файла <Ваше Имя>3.txt: «Сегодня: » и текущая дата.

16. Введите команду «cat <Ваше Имя>1.txt <Ваше Имя>3.txt > <Ваше Имя>5.txt». Например, «cat Иван1.txt Иван3.txt > Иван5.txt».

На основании файлов <Ваше Имя>1.txt и <Ваше Имя>3.txt будет создан новый файл <Ваше Имя>5.txt.

17. Просмотрите содержимое файла <Ваше Имя>5.txt.

18. Введите команду «cat <Ваше Имя>5.txt > <Ваше Имя>7.txt». Например, «cat Иван5.txt > Иван7.txt».

Файл <Ваше Имя>5.txt будет скопирован в файл <Ваше Имя>7.txt.

19. Просмотрите содержимое файла <Ваше Имя>7.txt.

20. Введите команду «cat /etc/man.conf».

Содержимое файла /etc/man.conf мгновенно промелькнет по экрану и пользователь увидит только последние строчки – с помощью команды «cat» можно просматривать только небольшие файлы. Для просмотра больших файлов удобнее пользоваться командами «more» или «less».

21. Введите команду «more /etc/man.conf».

Попытайтесь «полистать» файл /etc/man.conf вперед и назад. Нажмите клавишу «**?**» или «**h**», чтобы подробнее узнать возможности команды «more». Закончить просмотр можно по нажатию клавиши «**Q**». Более удобно просматривать файлы с помощью команды «less».

22. Введите команду «less /etc/man.conf».

Полистайте файл /etc/man.conf вперед и назад, ознакомьтесь с его содержанием. Нажмите клавишу «**h**», чтобы подробнее узнать возможности команды «less».

Шаг 5. Копирование и перемещение

Для практического освоения приемов копирования, перемещения и переименования выполните следующее:

1. Создайте каталог temp.
2. Сделайте его текущим.
3. Введите команду «cp ../<Ваше Имя>5.txt .». Например, «cp ../Иван5.txt .».

Обратите внимание на символ точки «.» в конце команды – это ссылка на текущий каталог, т. е. файл <Ваше Имя>5.txt из родительского каталога («..») будет скопирован в текущий каталог.

4. Введите команду «cal >> <Ваше Имя>5.txt». Например, «cal >> Иван5.txt».

5. Просмотрите содержимое файла <Ваше Имя>5.txt.

6. Перейдите в родительский каталог.

7. Просмотрите содержимое файла <Ваше Имя>5.txt.

8. Введите текст «cd te» и нажмите клавишу **Tab**.

В результате получится завершенная команда «cd temp/» – система сама умеет достраивать имена файлов и каталогов: пользователю достаточно набрать несколько первых символов имени файла или каталога и нажать клавишу **Tab**.

9. Нажмите клавишу **Enter**, чтобы выполнить построенную команду «cd temp/».

10. Введите команду «mv ../<Ваше Имя>7.txt .». Например, «mv ../Иван7.txt .».

11. Введите команду «ls ..».

В родительском каталоге файла <Ваше Имя>7.txt нет, …

12. Введите команду «ls».

… он был перенес в текущий каталог.

13. Переименуйте файл <Ваше Имя>7.txt в <Ваше Имя>9.txt.

14. Посмотрите содержимое текущего каталога.

В нем должен быть файл <Ваше Имя>9.txt вместо <Ваше Имя>7.txt.

Шаг 6. Сравнение файлов и заплатки

Для практического освоения приемов сравнения файлов и применения заплаток выполните следующее:

1. Прейдите в родительский каталог.

2. Создайте каталог diff-path.

3. Сделайте его текущим.

4. Выполните команду «cp ../temp/* .».

В результате из каталога temp в текущий каталог (diff-patch) будет скопировано все его содержание (символ «*» означает любые имена) – два файла: <Ваше Имя>5.txt и <Ваше Имя>9.txt.

5. Посмотрите содержание файлов <Ваше Имя>5.txt и <Ваше Имя>9.txt.

6. Введите команду «diff <Ваше Имя>9.txt <Ваше Имя>5.txt > <Ваше Имя>9-5.txt.diff». Например, «diff Иван9.txt Иван5.txt > Иван9-5.txt.diff».

Результат сравнения файлов <Ваше Имя>9.txt и <Ваше Имя>5.txt будет записан в файл <Ваше Имя>9-5.txt.diff.

7. Посмотрите содержание файла <Ваше Имя>9-5.txt.diff – отличие файла <Ваше Имя>9.txt от <Ваше Имя>5.txt.

8. Введите команду «patch <Ваше Имя>9.txt <Ваше Имя>9-5.txt.diff». Например, «patch Иван9.txt Иван9-5.txt.diff».

На основании «заплатки» – файла <Ваше Имя>9-5.diff.txt – содержимое файла <Ваше Имя>9.txt обновится и станет таким же, как и <Ваше Имя>5.txt.

9. Посмотрите содержание файлов <Ваше Имя>5.txt и <Ваше Имя>9.txt.

10. Введите команду «diff <Ваше Имя>9.txt <Ваше Имя>5.txt». Например, «diff Иван9.txt Иван5.txt».

Различий в файлах не обнаружено.

Шаг 7. Исследование прав доступа

Для практического освоения приемов изменения и использования прав доступа к файлам и каталогам выполните следующее:

1. Перейдите каталог <Ваша Фамилия>.
2. Создайте каталог access.
3. Выполните команду «ls –dl access».

Посмотрите права доступа вновь созданного каталога.

4. Перейдите в каталог access.
5. Скопируйте из каталога temp в текущий каталог (access) все его содержание.

Подсказка: Вы это уже делали это в Шаге «Сравнение файлов и заплатки».

6. Введите команду «cat <Ваше Имя>9.txt». Например, «cat Иван9.txt».

Посмотрите содержание файла <Ваше Имя>9.txt.

7. Введите команду «ls -l <Ваше Имя>9.txt». Например, «ls -l Иван9.txt».

Посмотрите права доступа к файлу <Ваше Имя>9.txt.

11. Введите команду «chmod -r <Ваше Имя>9.txt». Например, «chmod -r Иван9.txt».

В результате установили запрет на чтение всем категориям пользователей для файла <Ваше Имя>9.txt.

12. Посмотрите права доступа к файлу <Ваше Имя>9.txt.

13. Посмотрите содержание файла <Ваше Имя>9.txt.

В результате получите отказ в доступе – нет права на чтение.

14. Введите команду «date -u >> <Ваше Имя>9.txt». Например, «date –u >> Иван9.txt».

Дописали в конец файла <Ваше Имя>9.txt текущую дату с ключом «-u» – запись в защищенный на чтение файл возможна.

15. Введите команду «chmod +r <Ваше Имя>9.txt ; chmod -w <Ваше Имя>9.txt». Например, «chmod +r Иван9.txt ; chmod -w Иван9.txt».

Для файла <Ваше Имя>9.txt доступ на чтение разрешили («+r»), а доступ на запись запретили («-w»). Символ «;» позволяет в одной строке задать (перечислить) несколько команд.

16. Посмотрите права доступа к файлу <Ваше Имя>9.txt.

17. Введите команду «cal >> <Ваше Имя>9.txt». Например, «cal >> Иван9.txt».

В результате получим отказ в доступе – нет права на запись.

18. Введите команду «cat <Ваше Имя>9.txt». Например, «cat Иван9.txt».

А читать файл <Ваше Имя>9.txt можно.

19. Введите команду «rm <Ваше Имя>9.txt». Например, «rm Иван9.txt».

Удаление файла <Ваше Имя>9.txt сработает, хотя команда «rm», на всякий случай, и предупредит о том, что файл защищен от записи – «rm: удалить защищенный от записи обычный файл `<Ваше Имя>9.txt'?»

20. Подтвердите удаление файла, введя символ «**y**» (от англ. «yes») и нажав клавишу **Enter**.

21. Посмотрите права доступа к файлу <Ваше Имя>5.txt.

22. Введите команду «chmod -w .».

В результате запретили запись в текущий каталог.

23. Введите команду «rm <Ваше Имя>5.txt». Например, «rm Иван5.txt».

Сообщение «rm: невозможно удалить `<Ваше Имя>5.txt': Отказано в доступе» говорит о том, что даже не защищенный от записи файл нельзя удалить в защищенном от записи каталоге.

Шаг 8. Работа со ссылками

Для практического освоения приемов работы с жесткими и символическими ссылками выполните следующее:

1. Перейдите в каталог <Ваша Фамилия>.
2. Создайте каталог links и перейдите в него.
3. Скопируйте из каталога temp в текущий каталог (links) все его содержание.
4. Введите команду «ln <Ваше Имя>5.txt <Ваше Имя>5.txt.hardlink». Например, «ln Иван5.txt Иван5.txt.hardlink».

В результате будет создана вторая жесткая ссылка на файл <Ваше Имя>5.txt.

5. Введите команду «ls -li».

В левом столбике выведены номера индексных дескрипторов. Для файлов <Ваше Имя>5.txt и <Ваше Имя>5.txt.hardlink он один и тот же. У этих файлов одинаковые и остальные параметры: владелец, группа, размер и т. д. Одинаков у них также и третий столбик слева – количество жестких ссылок (2). Т. е. эти файлы имеют общую область хранения (и одно и то же содержание).

6. Посмотрите содержание файлов <Ваше Имя>5.txt и <Ваше Имя>5.txt.hardlink.
7. Введите команду «date >> <Ваше Имя>5.txt». Например, «date >> Иван5.txt».
8. Снова посмотрите содержание файлов <Ваше Имя>5.txt и <Ваше Имя>5.txt.hardlink.

Они обновились синхронно.

9. Введите команду «chmod -w <Ваше Имя>5.txt.hardlink». Например, «chmod -w Иван5.txt.hardlink».

10. Введите команду «ls -li».

Обратите внимание на поле прав доступа у файлов <Ваше Имя>5.txt и <Ваше Имя>5.txt.hardlink.

11. Снова введите команду «date >> <Ваше Имя>5.txt». Например, «date >> Иван5.txt».

В результате будет выдан отказ в доступе на запись для файла <Ваше Имя>5.txt, хотя изменяли права доступа для файла <Ваше Имя>5.txt.hardlink.

12. Введите команду «rm <Ваше Имя>5.txt.hardlink». Например, «rm Иван5.txt.hardlink».

13. Посмотрите количество жестких ссылок для файла <Ваше Имя>5.txt.

14. Введите команду «ln -s <Ваше Имя>9.txt <Ваше Имя>9.txt.symlink». Например, «ln -s Иван9.txt Иван9.txt.symlink».

В результате будет создана символическая ссылка на файл <Ваше Имя>9.txt.

15. Введите команду «ls -li».

Появился новый файл <Ваше Имя>9.txt.symlink типа «l» – символическая ссылка. Самое правое поле – имя – указывает («->»), на какой файл ссылается ссылка – <Ваше Имя>9.txt.

16. Посмотрите содержание файлов <Ваше Имя>9.txt и <Ваше Имя>9.txt.symlink.

17. Введите команду «cal >> <Ваше Имя>9.txt.symlink». Например, «cal >> Иван9.txt.symlink».

18. Снова посмотрите содержание файлов <Ваше Имя>9.txt и <Ваше Имя>9.txt.symlink.

Они имеют одно и то же содержание.

19. Введите команду «rm <Ваше Имя>9.txt». Например, «rm Иван9.txt».

20. Введите команду «ls -li».

Ссылка <Ваше Имя>9.txt.symlink изменила цвет – указывает на несуществующий файл.

21. Введите команду «cat <Ваше Имя>9.txt.symlink». Например, «cat Иван9.txt.symlink».

Будет выдано диагностическое сообщение, указывающее на то, что по ссылке файл не найден.

22. Введите команду «ln -s .. <Ваша Фамилия>». Например, «ln -s .. Иванов».

23. Введите команду «ls -l <Ваша Фамилия>». Например, «ls -l Иванов».

Посмотрите, куда указывает ссылка.

24. Введите команду «cd <Ваша Фамилия>». Например, «cd Иванов».

Шаг 9. Удаление

Для практического освоения приемов удаления файлов и каталогов выполните следующее:

1. Введите команду «cd ~/<Ваша Фамилия>». Например, «cd ~/Иванов».

Дабы из любого места файловой системы попасть в каталог <Ваша Фамилия>, если он почему-либо к настоящему моменту не был текущим.

2. Введите команду «rm links».

Будет выдано сообщение о том, что с помощью команды «rm» каталог так просто удалить нельзя.

3. Введите команду «rmdir links».

Еще одна неудача – удаляемый с помощью команды «rmdir» каталог должен быть пустым.

4. Перейдите в каталог links.
5. Введите команду «rm *».

В результате будут удалены все, имеющиеся в каталоге links, файлы. Обратите внимание на использование символа подстановки «*».

6. Посмотрите содержание текущего каталога.

Он должен быть пустым.

7. Введите команду «rm .».

Будет выдано диагностическое сообщение, говорящее о том, что «нельзя рубить сук, на котором сидишь».

8. Перейдите в родительский каталог.
9. Снова введите команду «rmdir links».

Теперь каталог links «молча» удалился.

10. Посмотрите содержимое текущего каталога.
11. Введите команду «rm –r access».

Ключ «–r» в команде «rm» говорит о том, что необходимо рекурсивно удалить все содержимое каталога access, а затем – и его. Но, попытка не увенчалась успехом – защищенный от записи каталог не так-то просто «разрушить».

12. Посмотрите права доступа для каталога access.
13. Установите для каталога access разрешение на запись.
14. Снова введите команду «rm –r access».

Ура! Все получилось! Каталог access уничтожен!

15. Посмотрите содержимое текущего каталога.
16. Самостоятельно удалите все вложенные подкаталоги каталога <Ваша Фамилия>, кроме каталога temp.

Индивидуальные задания

1. В каталоге ~/<Ваша Фамилия> построить дерево каталогов в соответствии с выбранным вариантом (см. далее).

2. В каталоге выделенным **полужирным** начертанием (например, **Каталог11**) создать текстовый файл под именем <Ваше имя>.txt (например, Иван.txt), в который записать свою фамилию, имя и группу.

3. Скопировать созданный ранее текстовый файл в каталог, выделенный начертанием ***полужирный курсив*** (например, ***Каталог132***).

4. Переименовать копию файла в <Ваше имя>1.txt.

5. В тетрадь для выполнения лабораторных работ занести полученные результаты: перечислить команды, которые использовались в работе.

6. Показать работу преподавателю.

7. Удалить, созданные в ходе выполнения лабораторной работы, файлы и каталоги.

Варианты деревьев каталогов

1.
```
──Каталог1──┬─Каталог11──Каталог111
            ├─Каталог12
            └─Каталог13─┬─Каталог131
                        └─Каталог132
```

2.
```
──┬─Directory1
  ├─Directory2
  └─Directory3──┬─Directory31
                ├─Directory32
                └─Directory33
```

3.
```
──┬─Каталог1
  ├─Каталог2
  └─Каталог3──┬─Каталог31
              └─Каталог32
```

4.
```
──Directory1──┬─Directory11──┬─Directory111
              └─Directory12  ├─Directory112
                             ├─Directory113
                             └─Directory114
```

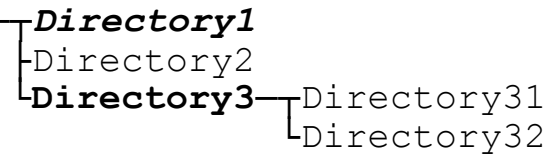
5.
```
──┬─Каталог1──┬─Каталог11
  └─Каталог2  ├─Каталог12──┬─Каталог121
              └─Каталог21  └─Каталог123
```

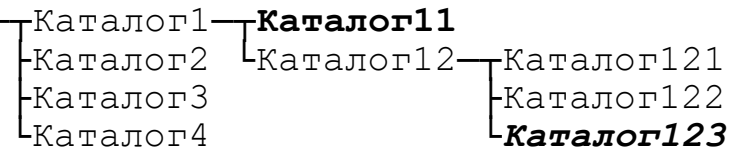
6.
```
──┬─Directory1──┬─Directory11──┬─Directory111
  ├─Directory2  ├─Directory12  ├─Folde112
  └─Directory3  └─Directory13  └─Directory113
```

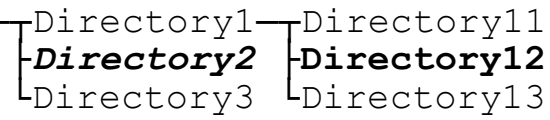
7.
```
──┬─Каталог1──┬─Каталог11
  ├─Каталог2  ├─Каталог12
  └─Каталог3  └─Каталог13
```

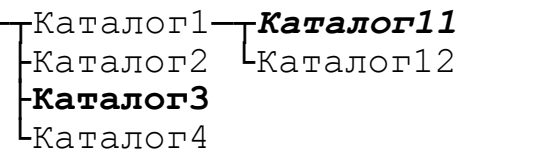
8.
```
──┬─Directory1  ┌─Directory31
  ├─Directory2  ├─Directory32
  └─Directory3──└─Directory33
```

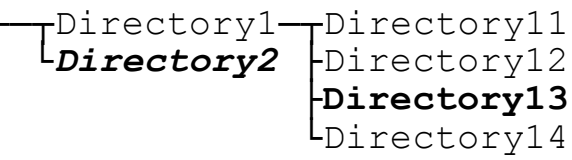
9.
```
──┬─Каталог1──┬─Каталог11
  │           └─Каталог12
  └─Каталог2──┬─Каталог21
              └─Каталог22
```

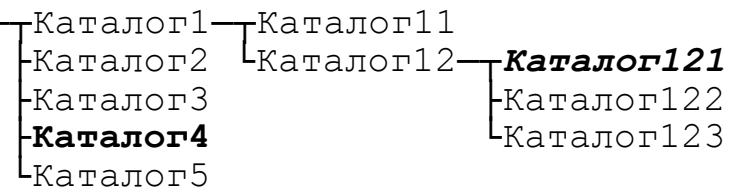
10.


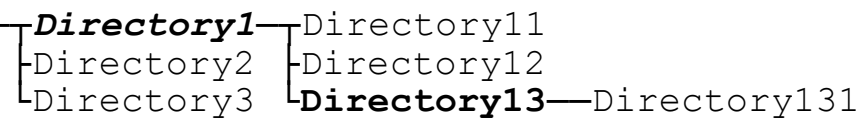
11.


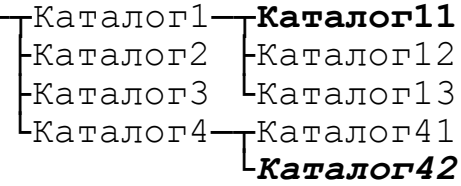
12.


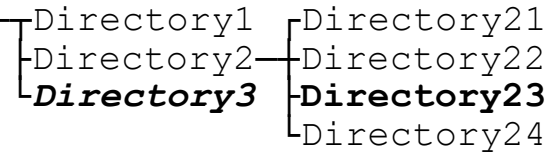
13.


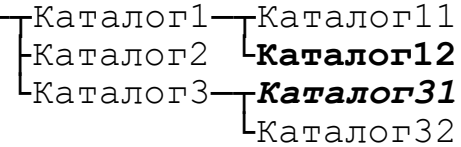
14.


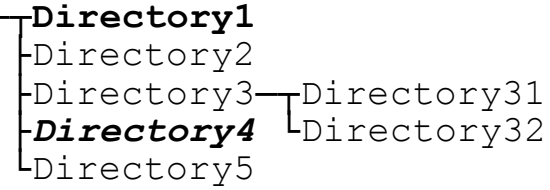
15.


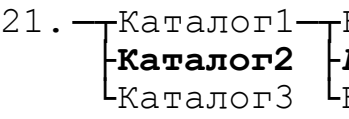
16.


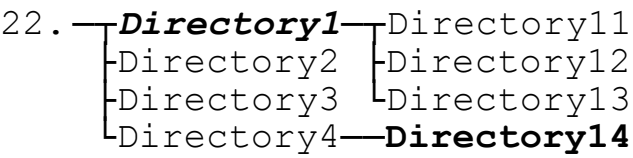
17.


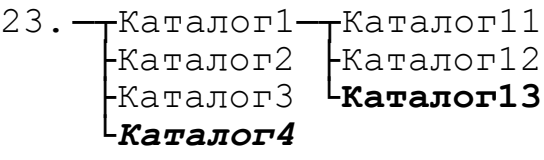
18.


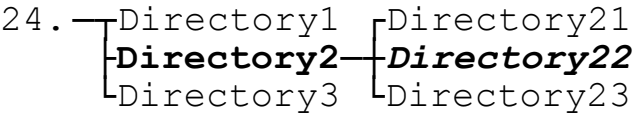
19.


20.


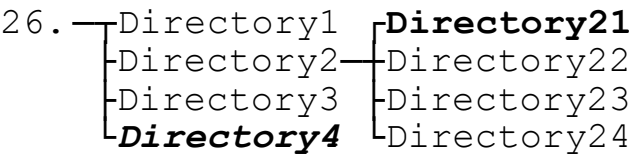
21.


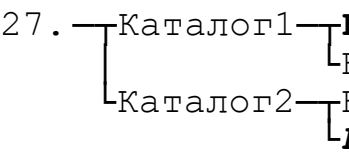
22.


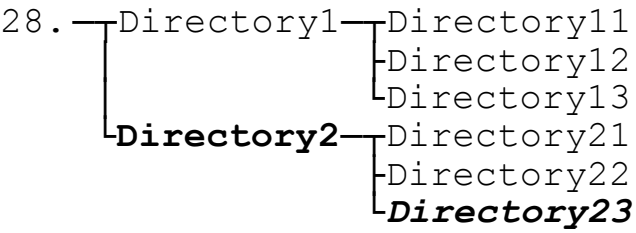
23.


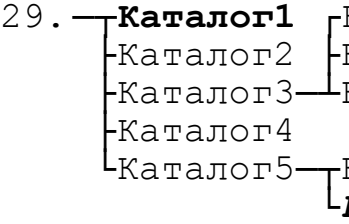
24.


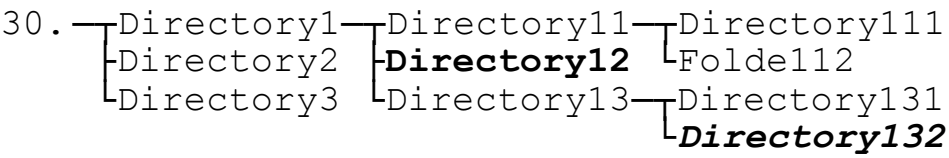
25.


26.


27.


28.


29.


30.


Контрольные вопросы

1. Что такое файловая система?
2. Что такое название файловой системы?
3. Какие файловые системы являются основными для ОС Linux?
4. Назовите файловые системы, с которыми может работать ОС Linux.
5. Как называется та часть операционной системы, которая обеспечивает работу файловых систем?
6. С какими объектами работает файловая система Linux?
7. Что такое файл?
8. Назовите основную характеристику файла с точки зрения пользователя.
9. По каким правилам строится имя файла?
10. Что такое индексный дескриптор?
11. Что такое жесткая ссылка? Что такое символическая ссылка? Что у них общего, и чем они отличается?
12. Что такое каталог? Зачем он нужен?
13. Как подключить и отключить файловую систему на другом носителе в общее дерево каталогов ОС Linux?
14. По каким правилам строится имя каталога?
15. Что такое текущий каталог? Что такое домашний каталог?
16. Как, и где, можно посмотреть имя текущего каталога?
17. Как сменить текущий каталог?
18. Как посмотреть содержимое каталога? Какую информацию при этом можно увидеть?
19. Что такое FHS? Назовите основные (под)каталоги корневого каталога ОС Linux.
20. Назовите типы файлов, которые поддерживает Linux. Как можно посмотреть тип файла?
21. Как организовано разграничение доступа к файлам и каталогам в Linux? Как посмотреть права доступа к файлу?
22. Как изменить права доступа к файлу? Кто может их изменять?
23. Как создать каталог? Что значат имена: «`.`» и «`..`»?
24. Как удалить файл, каталог?
25. Как скопировать файл, каталог?
26. Как переместить, переименовать файл, каталог?
27. Для чего служит (что может) команда «`cat`»?
28. Как просмотреть файл, который не помещается на экране монитора?
29. Как найти файл или папку? Что такое символы шаблона, как они используются?
30. Как сравнить два файла? Как автоматически откорректировать более старую версию файла?
31. Сформулируйте и задайте свой вопрос по пройденному материалу.
32. Сформулируйте и внесите свое предложение по совершенствованию темы.

Лабораторная работа № 3
Командная оболочка Shell

Цель работы

Освоение и приобретение практических навыков работы с командной оболочкой ОС Linux.

Краткие теоретические сведения

Общение пользователя с операционной системой Linux в текстовом режиме происходит через так называемую «*командную оболочку*» («*оболочку командной строки*», «*оболочку*» или, просто, «*shell*»). В терминах других операционных систем эта компонента называется еще «*интерпретатором команд*» или «*командным процессором*». Но в Linux (а также, других UNIX-подобных ОС) не совсем благозвучное название «*оболочка*» очень точно отражает суть проблемы – пользователь работает не с ОС непосредственно, а с ОС через специальную программу-оболочку – shell. Т. е., все, что вводит пользователь на виртуальном терминале, вначале обрабатывается оболочкой. Далее, она вызывает на выполнение необходимые программы, а результаты их работы возвращает, опять же, пользователю на виртуальный терминал.

Имя программного файла одной из первых оболочек операционной системы UNIX было «`sh`» (сокращение от shell). Потом были разработаны несколько ее улучшенных вариантов. В частности, *Bourne shell* – расширенная версия `sh`, написанная Стивом Борном (Steve Bourne). В рамках проекта GNU Брайеном Фоксом (Brain Fox – основной разработчик) и Четом Рэми (Chet Ramey) была разработана оболочка `bash` (от англ. «Bourne again shell» – «снова оболочка Борна» или «заново рожденная shell»).

На сегодняшний день разнообразие названий командных оболочек в мире UNIX может привести в замешательство кого угодно. Но, к этому нужно относиться как к результату эволюционного развития. На самом деле, существует только две основные группы (два основных типа) оболочек, это:

- *Bourne shell* и
- *C shell*.

Самая новая из командных оболочек, совместимая с оболочкой Борна – это *оболочка Z* (`zsh`). Оболочки группы C shell (`csh` и `tcsh`) используют синтаксис, чем-то напоминающий синтаксис языка программирования Си.

Выбор типа оболочки – это почти как выбор религии. Кто-то предпочитает синтаксис shell Борна, а кто-то – более структурированный синтаксис C shell. Однако, с точки зрения рядового пользователя, различия между всеми этими оболочками практически отсутствуют. Такие команды, например, как «`cp` и «`ls`», работают совершенно одинаково во всех типах оболочек. Различия начинают проявляться только при написании командных файлов, или использовании некоторых новых свойств оболочек. Однако, если есть такая возможность, попробуйте поработать с несколькими оболочками, прежде чем

остановиться на одной из них. Это будет полезно на тот случай, если когда-нибудь Вы столкнетесь с системой, где выбор оболочек будет ограничен.

Самой же распространенной оболочкой, на сегодняшний день, в большинстве дистрибутивов Linux является `bash` – ее и будем использовать. А термином shell обозначают, вообще, любую командную оболочку, в любой UNIX-подобной ОС.

Определить, какая оболочка запущена, можно с помощью команды «`echo $SHELL`». Команда «`echo`» отображает значение переменной окружения `SHELL`, в которой записано полное имя файла командной оболочки. Переменные окружения служат для задания параметров работы операционной системы. В ОС Linux таких переменных окружения достаточно много. Их имена задаются, как правило, прописными буквами. А посмотреть значение такой переменной можно при помощи команды «`echo`» (только не пропускайте знак доллара «`$`» перед именем переменной!). Команда «`set`» без параметров отображает текущие значения всех переменных окружения shell.

Управление работой Linux в текстовом режиме осуществляется при помощи команд. Т. е., пользователь вводит команду, ОС ее выполняет, возвращает результат и далее – ждет ввода следующей команды (чаще всего, выдавая символ приглашения «`$`»). Ввод каждой команды завершается нажатием клавиши **`Enter`**. Формально, синтаксис команды Linux выглядит следующим образом:

```
<имя команды> [<ключи>] [<параметры>].
```

Т. е., каждая команда обязательно имеет *имя*, например, «`ls`» – вывести содержимое каталога. Она, также, может иметь (а может, и не меть) *параметры*, которые уточняют способ ее выполнения. Например, в команде «`ls`», имя каталога, содержимое которого необходимо отобразить. Команда может содержать еще и *ключи* (*опции*, *флаги*), которые еще более точно определяют способ ее выполнения. Например, ключ «`-l`» команды `ls` вызывает вывод информации о содержании каталога в расширенном (длинном) формате. Ключи в команде обязательно начинаются символом «тире» («`-`»). Однобуквенные ключи можно объединять. Компоненты команды (имя, ключи и параметры) друг от друга отделяются, по крайней мере, одним пробелом.

В ОС Linux имеется масса разнообразных команд. При этом можно также добавлять новые команды, или изменять уже имеющиеся. Это в корне отличает UNIX-подобные ОС от большинства других операционных систем, которые содержат строго ограниченный набор команд.

Все команды, которые «понимает» Linux, делятся на две группы:

1. внутренние, и
2. внешние.

Внутренние команды встроены («вшиты», «зашиты», «запрограммированы») непосредственно в оболочку. Это, например, такие команды как «`cd`», «`echo`» или «`exit`». Внешние команды – *утилиты* – это просто файлы. Например, команда «`ls`» – это исполняемый файл,

расположенный в каталоге /bin. Поэтому вместо команды «ls» можно ввести полное имя исполняемого файла – «/bin/ls».

Список встроенных команд shell можно посмотреть по команде «help». Детальную информацию по конкретной встроенной команде выдает та же команда «help» с указанием в качестве параметра имени встроенной команды, например: «help cd».

Тип любой команды shell можно определить с помощью команды «type». Встроенные команды называются «builtin», а для внешних команд выводится полное имя исполнительного файла. Например, команды «type man» и «type echo» выдадут, соответственно:

man is /usr/bin/man и
echo is a shell builtin.

Некоторые – самые нужные – команды бывают, и встроенными, и в виде отдельного файла. Например, «echo». Работает встроенная команда так же, как и внешняя, но так как времени на ее выполнение уходит существенно меньше, командный интерпретатор выберет именно ее, если будет такая возможность. Ключ «-a» заставляет «type» вывести все возможные варианты интерпретации команды, а ключ «-t» – вывести тип команды вместо полного имени. Например, команды «type -a echo» и «type -at echo» выдадут, соответственно:

echo is a shell builtin
echo is /bin/echo и
и
builtin
file.

Именно набор внешних команд можно расширять и изменять. Благодаря этому Linux является очень мощной и гибкой системой. Чтобы предоставить новую команду, системный администратор может просто установить ее в стандартный каталог, где размещаются программы. Список каталогов, где оболочка отыскивает команды, называется «*путь поиска*». Он хранится в переменной окружения PATH. Посмотреть ее текущее значение можно по команде «echo $PATH». Результатом работы команды будет последовательность путей, разделенных символом двоеточия («:»), например:

/home/student21/bin:/bin:/usr/bin:/usr/local/bin:
/usr/X11R6/bin:/usr/games.

Здесь, первый путь – /home/student21/bin, второй – /bin, и т. д. Поэтому, если существуют две команды с одинаковыми именами, одна из которых находится в каталоге /home/student21/bin, а другая в каталоге /bin, то выполнится та, которая находится в каталоге /home/student21/bin.

Большинство современных командных оболочек (за исключением самых ранних `sh` и `csh`) допускают редактирование вводимой информации (командной строки). Возможности редактора командной строки специфичны для разных командных оболочек. Однако самые необходимые команды редактирования поддерживаются во всех разновидностях shell сходным образом. Так, например:

- **Backspace** – удалить символ слева от курсора,
- **Delete** – удалить символ в позиции от курсора,
- **Ctrl+U** – удалить подстроку от курсора до начала строки,
- **Ctrl+K** – удалить подстроку от курсора до конца строки,
- ← «стрелка влево» – переместить курсор на одну позицию влево,
- → «стрелка вправо» – переместить курсор на одну позицию вправо,
- **Home** или **Ctrl+A** – переместить курсор в начало командной строки,
- **End** или **Ctrl+E** – переместить курсор в конец командной строки,
- **Alt+B** – переместить курсор на одно слово влево,
- **Alt+F** – переместить курсор на одно слово вправо,
- **Ctrl+W** – удалить слово слева от курсора,
- **Alt+T** – поменять местами символ в позиции курсора и слева от него.

Нажатие клавиш **Ctrl+C** вызывает немедленное прекращение выполнения команды, а **Ctrl+Z** – приостанов. Комбинация **Ctrl+S** останавливает вывод на экран, пока он не будет продолжен с помощью **Ctrl+Q**.

Изменить стандартные назначения комбинаций клавиш можно командой «`stty`», используя следующий синтаксис:

`stty <функция> <сочетание_клавиш>.`

Например, команда «`stty erase ^H`» назначает удаление символа слева от курсора сочетанию клавиш **Ctrl+H** – то, что, обычно, делает клавиша **Backspace**. Диакритический знак «крышечка» («^») перед клавишей «**H**» обозначает клавишу **Ctrl** – это общепринятая нотация во всех UNIX-системах.

Список текущих установок терминала можно получить, введя команду «`stty -a`». Но разобраться в результатах ее работы не так то просто: «`stty`» – сложная команда, имеющая многочисленные применения, и для некоторых из них требуются обширные познания. Более подробную информацию по команде «`stty`» можно получить, выполнив команду «`man stty`».

Если по какой либо причине настройки текстового виртуального терминала собьются (такое может произойти, например, при попытке просмотреть содержимое двоичного (исполняемого) файла с помощью команды «`cat`») и он находится в некотором непонятном состоянии – не реагирует на **Enter**, не показывает ввода, не удаляет символы, выводит текст «ступеньками» и т. п., то «лечить» его рекомендуется при помощи следующих команд:

- «`stty sane`» – специальная форма «`stty`», сбрасывающая настройки терминала в пригодное для работы состояние,

• «`setterm -reset`» – вернет настройки терминала в обычное состояние,

• «`tput reset`», «`reset`» или «`tset`» – инициализации терминала.

Все современные командные оболочки, в большей или меньшей степени, умеют достраивать (англ. «completion») имена файлов и команд. Для этого необходимо набрать несколько первых символов имени и нажать клавишу **Tab**. Например, введите следующие символы «`cd /usr/inc`» и, не нажимая клавишу **Enter**, нажмите клавишу **Tab**. Оболочка сама добавит «`lude/`», дополнив строку до полного имени каталога «`/usr/include/`». Теперь можно нажать **Enter**, и команда будет выполнена.

Критерием определения имени файла является минимальная полнота. Введите символы в количестве, достаточном, чтобы по ним можно было отличить требуемое имя файла или каталога от всех других в этом каталоге. Оболочка сможет отыскать это имя и дополнить его – вплоть до символа слэш, если найденное имя является именем каталога. Функция автодополнения может использоваться не только с именами файлов и каталогов, но и с командами – если это слово стоит в начале командной строки.

Выполняя достраивание, shell может вывести не всю строку, а только ту ее часть, относительно которой нет сомнений. Если дальнейшее достраивание может пойти несколькими путями, то однократное нажатие **Tab** ни к чему не приведет, а повторное – к выводу над командной строкой списка всех возможных вариантов. В этом случае надо подсказать командной оболочке продолжение: дописать несколько символов, определяющих, по какому пути пойдет достраивание, и снова нажать **Tab**. Если дважды нажать клавишу **Tab** в пустой командной строке, то будет выдано, примерно, такое сообщение: «`Display all 2148 possibilities? (y or n)`» – «`Показать все 2148 возможности? (y или n)`». Аналогичным образом можно пытаться «угадывать» окончания команд, если вместо клавиши **Tab** несколько раз нажимать клавишу **Esc**.

Все команды, которые вводил пользователь, сохраняются в так называемой «*истории команд*». Наиболее просто «извлечь» их оттуда можно при помощи стрелок управления курсором:

• «стрелка вверх» (↑) или **Ctrl+P** (от англ. «previous» – «предыдущий») «прокручивают» историю команд от последней к первой, а

• «стрелка вниз» (↓) или **Ctrl+N** (от англ. «next» – «потом») – в обратном направлении.

Соответствующая команда отображается в командной строке, как только что набранная. Ее можно отредактировать, или выполнить непосредственно, нажав клавишу **Enter**. Максимальное количество команд, которое может храниться в истории, содержит переменная окружения `HISTSIZE`.

Если необходимо найти в истории какую-то определенную команду, проще не «листать» список истории стрелками, а поискать в ней с помощью сочетания клавиш **Ctrl+R** (от англ. «reverse search» – «обратный поиск»). При этом выводится подсказка специального вида – «`(reverse-i-search)`**`':`**».

Нажимая символы для поиска, находим первую команду, в которой присутствует введенная подстрока поиска. Далее, при помощи **Ctrl+R**, отыскиваются все остальные команды с заданной подстрокой.

Чтобы история команд могла сохраняться между сеансами работы пользователя, shell записывает ее в файл, имя которого содержится в переменной окружения `HISTFILE` (обычно, это `.bash_history`, находящийся в домашнем каталоге пользователя). Делается это в момент завершения сеанса работы пользователя – накопленная за время работы история дописывается в конец этого файла. При следующем запуске shell этот файл считывает целиком. История хранится не вечно – количество запоминаемых команд ограничено – обычно, около 10000 команд. Точное значение этой величины хранится в переменной окружения `HISTFILESIZE`.

Для просмотра всей истории команд служит команда «`history`». Она выводит информацию в две колонки: номер команды и саму команду. «Полистать» несколько последних страниц вывода команды «`history`» можно при помощи сочетаний клавиш **Shift+PgUp** и **Shift+PgDown**.

Подавляющее большинство команд Linux, по умолчанию, данные вводят с клавиатуры, а результат отображают на экране монитора. При этом клавиатура является «*стандартным устройством ввода*» (`stdin`), а экран виртуального терминала – «*стандартным устройством вывода*» (`stdout`). Однако, поскольку Linux – система гибкая, эти настройки по умолчанию можно динамически изменять по ходу выполнения команд.

Например, shell позволяет свободно перенаправлять вывод любой команды с терминала в файл. Для этого необходимо после команды добавить знак «больше чем» («>»), а вслед за ним указать имя файла, куда будет послан вывод команды. Например, чтобы перехватить и сохранить результаты работы команды «`ls`», можно ввести:

```
ls /usr/bin > ~/dir.txt
```

Перечень файлов, находящихся в каталоге `/usr/bin`, будет записан в файл `dir.txt` в домашнем каталоге пользователя. Если файл `dir.txt` уже существует, то операция перенаправления вывода («>») затрет его старое содержимое. Перезапись существующего файла является частой ошибкой пользователей. Поэтому, чтобы продолжить записывать (дописывать) данные в уже существующий файл, в shell имеется другая операция перенаправления вывода, а именно – два знака «больше чем» («>>»). Например, команда:

```
date >> dir.txt
```

добавит в конец уже существующего файла `dir.txt` данные со стандартного вывода очередной команды `date` – текущую дату – будет получен именно тот результат, который требовался. Прием перенаправления вывода оказывается очень удобным, когда нужно неоднократно выполнить одну и ту же программу и сохранять результаты ее работы, например, для анализа ошибок.

Однако, нередко возникают ситуации, когда необходимо, чтобы одна программа обрабатывала данные, выводимые другой программой. Пользуясь перенаправлением ввода-вывода, можно сохранить вывод одной программы в файле, а потом направить этот файл на ввод другой программе. Но то же самое можно сделать и более эффективно – перенаправлять вывод можно не только в файл, но и непосредственно на ввод другой программе. В этом случае вместо двух команд потребуется только одна – программы передают друг другу данные «из рук в руки». В Linux такой способ передачи данных называется «*программным каналом*» или «*конвейером*». В англоязычных, оригинальных, текстах термину «*конвейер*» соответствуют термины «*pipe*» – «труба» или, более точно, «pipeline» – «трубопровод».

В shell для перенаправления вывода одной программы на ввод другой программе служит символ «|». Самый простой и наиболее распространенный случай, когда требуется использовать конвейер, возникает, если вывод программы не умещается на экране монитора и очень быстро «пролетает» перед глазами, так что пользователь не успевает его прочитать. В этом случае можно направить вывод в программу просмотра, например, «`less`», которая позволит не торопясь, в свое удовольствие, «пролистать» весь текст, вернуться к началу и т. п.

```
ls -l /usr/bin | less
```

Можно последовательно обработать данные несколькими разными программами, перенаправляя вывод одной на ввод следующей программе и организуя, таким образом, сколь угодно длинный конвейер для получения необходимого результата. Т. е., программные каналы используются для того, чтобы скомбинировать несколько маленьких программ, каждая из которых выполняет только определенные преобразования над своим входным потоком, для создания обобщенной команды, результатом которой будет какое-то более сложное преобразование.

Необходимо иметь в виду, что перенаправления и конвейеры, это средства, предоставляемые оболочкой shell, это синтаксис shell, и символы «>», «>>» и «|» не имеют никакого отношения к командам, как таковым.

Помимо перенаправления вывода команды в файл, или на вход другой команды, в shell имеется еще один механизм передачи информации – вывод любой команды может быть помещен в командную строку. Для этого такая команда должна быть заключена в «одинарные обратные кавычки» – «`» (которые находятся на одной клавише с символом «тильда» – «~»). Например, команда «`echo Файлов: `ls -l /usr/bin | wc -l`` отображает слово «`Файлов: `», за которым будет число, на единицу больше количества файлов в каталоге `/usr/bin` (строка «итого»). Т. е., вывод команды, заключенной в одинарные обратные кавычки – «`ls -l /usr/bin | wc -l`» – был помещен в командную строку – дополнил параметр, и был выведен – результат работы команды «`echo`».

Однако, вводить такие длинные команды, да еще и не ошибаясь, довольно-таки трудно. А если делать это приходится многократно – то не только трудно, но и утомительно. Для этого в shell из длинных, и часто повторяющихся, «старых» команд создают «новые», более короткие, команды – просто записывают их в специальные файлы – «*сценарии*».

Существует большое семейство программ, смысл работы которых заключается в том, чтобы воспринимать входной поток данных, производить над ним некоторые преобразования и выдавать результат на стандартный вывод (откуда его можно перенаправить куда-то еще по желанию пользователя). В Linux такие программы называются «*фильтрами*»: данные проходят через них, причем что-то «застревает» в фильтре и не появляется на выходе, что-то изменяется, а что-то проходит сквозь фильтр неизменным. Фильтры в Linux, обычно, читают данные со стандартного ввода, а результат выводят на стандартный вывод.

К числу наиболее часто используемых программ-фильтров относятся, например, такие команды, как: «`cat`», «`ls`», «`cp`», «`more`», «`less`», «`wc`», «`cmp`», «`diff`», «`sort`» и др. Простейшим фильтром, безусловно, является команда «`echo`». Она предназначена для выдачи на стандартное устройство вывода строки символов, которая задана ей в качестве аргумента. После этого она выдает сигнал перевода строки и завершается. Например, команда «`echo Здравствуй, мир!`» в качестве результата своей работы вернет строку символов «`Здравствуй, мир!`».

Другим примером программы-фильтра является команда «`sort`», которая сортирует входные данные и посылает результат на стандартный выход. Кроме этого очевидного действия, выполняемого по умолчанию, есть множество других способов сортировки, для каждого из которых «`sort`» имеет соответствующий ключ. Например, ключ «`-f`» вызывает сортировку не чувствительной к регистру символов. Чаще всего сортировка выполняется по символам, но иногда требуется сортировать числа. Ключ «`-n`» обеспечивает сортировку по числовым значениям, а ключ «`-r`» изменяет порядок сортировки на противоположный. Таким образом, следующие команды сортируют содержимое каталога `/usr/bin`:

```
ls /usr/bin | sort -f          – в алфавитном порядке,
ls -s /usr/bin | sort -n       – по возрастанию размера,
ls -s /usr/bin | sort -nr      – по убыванию размера.
```

По умолчанию «`sort`» сравнивает строки целиком, но ей можно указать, по какому конкретному полю должна выполняться сортировка. Для этого предназначен ключ «`-k`». Запись «`-kn`» означает, что сортировка будет выполняться по полю номер `n`. Например, команда:

```
ls -la /usr/bin | sort -k5nr
```

выполняет сортировку содержимого каталога `/usr/bin` по убыванию размера файла.

Еще одна, часто используемая программа-фильтр – это команда «`wc`» (от англ. «word count» – «счетчик слов»), которая подсчитывает количество строк, слов и байт. Для подсчета только строк необходимо указать ключ «`-l`», слов – «`-w`», а байт – «`-c`». Например, команда «`wc -w Фрукты.txt`» выдаст число – количество слов, которое содержит файл `Фрукты.txt`.

Особым фильтром является команда «`tee`» («мишень»), которая «раздваивает» входной поток: с одной стороны – направляя его на стандартный вывод, а с другой – в файл, имя которого задано ей в качестве параметра.

Данные, проходящие через фильтр, представляют собой текст: в стандартных потоках ввода-вывода все данные передаются в виде символов, строка за строкой, как и в терминале. Поэтому при помощи конвейера могут быть состыкованы любые две программ, поддерживающих стандартные потоки ввода-вывода. Это, в какой-то степени, напоминает стандартный детский конструктор, где все детали могут состыковываться между собой.

В любом дистрибутиве Linux обязательно присутствует набор стандартных утилит, предназначенных для работы с файловой системой и обработки текста. Каждая из этих утилит предназначена для выполнения какой-то одной операции над файлами или текстом: вывод списка файлов в каталоге, копирование, сортировка строк и т. д. При этом каждая утилита может выполнять свою функцию несколько по-разному, в зависимости от переданных ей ключей и параметров. Все утилиты работают со стандартными потоками ввода-вывода, поэтому хорошо приспособлены для построения конвейеров: последовательно выполняя простые операции над потоком данных, можно решать довольно нетривиальные задачи.

Принцип комбинирования элементарных операций для выполнения сложных задач унаследован Linux-ом от операционной системы UNIX (как, впрочем, и многие другие принципы). Подавляющее большинство утилит UNIX, не потеряли своего значения и в Linux. Все они ориентированы на работу с данными в текстовой форме, многие являются фильтрами, все не имеют графического интерфейса и вызываются из командной строки. Этот пакет утилит называется «*coreutils*» – «*основные утилиты*».

Командная оболочка shell может использоваться не только для выполнения отдельных команд, но также и для автоматического исполнения некоторой серии таких команд с помощью «*сценария*». Т. е., если какую-то последовательность команд приходится выполнять достаточно часто, то ее можно превратить в «новую» команду, имеющую собственное имя так, чтобы ее можно было использовать в дальнейшем как обычную команду. Собранные в одном файле такие серии команд в англоязычной литературе последнее время преимущественно называются «*script*» – «*сценарий*», хотя то, что под этим понимается, во многих книгах на русском традиционно называется как «*программа на shell*», «*командный файл оболочки*» или, просто, «*командный файл*».

Сценарий – это обычный исполняемый текстовый файл, предназначенный для обработки какой-нибудь утилитой. Чаще всего такая утилита – это интерпретатор некоторого языка программирования, а содержимое такого

файла – программа на этом языке. Если системе специально не «намекнуть», то в качестве интерпретатора она запускает стандартный shell – /bin/sh.

Например, если создать текстовый файл Hello.script и записать в него строку «echo Здравствуй, Мир!», а затем запустить его на выполнение по команде «sh Hello.script» – будет выдано сообщение «Здравствуй, Мир!». Для того, чтобы всякий раз не приходилось явно указывать программу-интерпретатор (sh или bash) и передавать ей сценарий в виде параметра, а просто запускать скрипт на выполнение по имени файла (например, Hello.script), вне зависимости от того, в каком каталоге работает пользователь, необходимо:

1. при помощи команды «chmod» установить параметр доступа «исполняемый», поскольку по умолчанию текстовые файлы создаются доступным на запись и чтение, но не на выполнение (например, «chmod +x Hello.script») и

2. поместить его в один из каталогов поиска команд оболочки shell, которые перечисляются в переменной окружения PATH. Если скрипт предназначен не для общего применения, а для личного, то его лучше поместить в личный каталог программ (команд), который, обычно, находится в домашнем каталоге пользователя и называется bin. Такой каталог лучше всего записать первым в списке переменной окружения PATH, поскольку, если имя скрипта совпадет с именем одной из утилит в других каталогах поиска, то выполняться будет именно он, а не наоборот – команда из другого каталога.

«Правильный» скрипт командной оболочки shell в первой строке должен содержать имя программы-интерпретатора, которая будет его исполнять. Для этого первыми двумя байтами сценария должны быть символы «#!». Тогда всю его первую строку, начиная с третьего байта, система воспринимает как команду обработки сценария. Исполнение такого сценария приведет к запуску указанной после «#!» команды, последним параметром которой будет имя самого файла сценария. Однако, если между символами «#» и «!» окажется пробел, то данная директива не сработает, поскольку будет воспринята как обычный комментарий. Тогда содержание «правильного» сценария Hello.script должно быть таким:

```
#!/bin/sh
# Сценарий shell, который выводит сообщение
echo Здравствуй, Мир!
```

В нем первая строка «#!/bin/sh» говорит о том, что этот файл – сценарий, и сообщает shell, как его выполнить. В данном случае необходимо передать сценарий на выполнение команде /bin/sh, т. е. самой программе shell. Почему это важно? В большинстве систем UNIX /bin/sh/ – это shell Баурновского типа, например bash. При этом гарантируется, что сценарий будет выполняться именно под shell Баурновского типа, а не, скажем, под C shell. Т. е., этот сценарий будет выполняться под shell Баурна, даже в том случае, когда текущим интерпретатором команд является tcsh (или какой-

либо другой C shell). Все эти командные оболочки друг от друга отличаются только синтаксисом языка, на котором пишутся сценарии.

Вторая строка – это комментарий. Комментарии в скриптах начинаются символом «решетка» («#») и продолжаются до конца строки. Т. е., все, что находится в такой строке после символа «решетка» и до конца строки, shell будет считать комментарием и игнорировать – этот текст не рассматривается как команд, а используются лишь для пояснений.

Остальные строки сценария – обычные команды в том виде, в каком они вводятся прямо на выполнение. Интерпретатор читает каждую строку сценария и выполняет эту строку, как будто она введена в ответ на приглашение.

Однако, есть возможность написать сценарий не только для shell, а вообще, для любой утилиты, в том числе и написанной самостоятельно. Например, исполнение такого сценария:

```
#!/bin/sort
Яблоки
Груши
Сливы
```

приведет к запуску на выполнение указанной после «#!» команды `/bin/sort` и передачи ей в качестве последнего параметра самого сценария. Т е., запустится утилита «`sort`», а сценарий (файл с неотсортированными строками) передастся ей в качестве параметра. Конечно, оформлять файлы, которые необходимо отсортировать, в виде sort-сценариев довольно бессмысленно. Однако, данный пример продемонстрировал, что сценарий можно написать для всего, чего угодно. Для этого необходимо лишь заменить первую строку ссылкой на любую программу, которая будет читать файл, и исполнять соответствующие команды. Например, скрипты на языке Perl начинаются со строки вида: «`#!/usr/bin/perl`».

Самую полную информацию по работе с командной оболочкой `bash` можно получить по команде «`man bash`».

Все, с теорией закончили. Настала пора практики. В конце концов, учиться лучше всего на примерах. И дальше – по инструкции:

Ход работы

Шаг 1. Исследование основных параметров shell

Для практического ознакомления с основными параметрами shell выполните следующее:

1. Введите команду «`echo $SHELL`» (не забудьте про символ доллара «$» перед именем переменной окружения `SHELL`!).

В результате будет выдано полное имя файла запущенной командной оболочки.

2. Введите команду «`ls -l /bin/<имя файла оболочки>`». Например, «`ls -l /bin/bash`».

В результате будут выданы параметры исполняемого файла оболочки.

3. Введите команду «`ls -l /bin/*sh`».

В результате будут выданы все, установленные в системе, оболочки.

4. Введите команду «`echo $PATH`».

В результате будет выдана строка, примерно, такого вида:

```
/home/student21/bin:/bin:/usr/bin:/usr/local/bin:
            /usr/X11R6/bin:/usr/games
```

В ней приведен порядок просмотра каталогов (путей), который выполняет оболочка shell при поиске внешней команды.

Шаг 2. Исследование типа команд

Для практического исследования типа команд shell выполните следующее:

1. Введите команду «`help`».

В результате будет выдан список встроенных команд shell.

2. Введите команду «`help echo`».

В результате будет выдана справка по конкретной внутренней команде – «`echo`».

3. Введите команду «`type echo`».

В результате будет выдано сообщение: «`echo is a shell builtin`», говорящее о том, что команда «`echo`» внутренняя («`builtin`») команда shell.

4. Введите команду «`type man`».

В результате будет выдано сообщение: «`man is /usr/bin/man`» – полное имя исполнительного файла внешней команды (утилиты) «`man`».

5. Введите команду «`type -at echo`».

Оказывается, команда `echo` существует в 2-х вариантах: есть и встроенная, и внешняя.

6. Введите команду «`type -a pwd`».

Команда «`pwd`» – отобразить имя текущего (рабочего) каталога – тоже существует в двух вариантах.

7. Самостоятельно, отобразите параметры исполняемого файла команды «`echo`».

Шаг 3. Дополнение

Для практического освоения приемов дополнения (завершения) имен файлов, каталогов и команд выполните следующее:

1. Введите следующую последовательность символов «`cd /usr/inc`» и, не нажимая клавишу **`Enter`**, нажмите клавишу **`Tab`**.

Оболочка сама добавит «`lude/`», дополнив строку до полного имени каталога – «`/usr/include/`» – получится законченная команда: «`cd /usr/include/`». Теперь можно нажать **`Enter`**, и команда будет выполнена.

2. Введите следующие символы «`cd /usr/l`» и, не нажимая клавишу **`Enter`**, нажмите клавишу **`Tab`**.

Поскольку по одному символу «`l`» однозначно достроить имя файла или каталога не возможно, ничего выдано не будет.

3. Второй раз нажмите клавишу **`Tab`**.

Над командной строкой будет выдан списка всех возможных вариантов. Например:

```
lib/     libexec/     local/.
```

4. Введите символ «o» и нажмите клавишу **Tab** а, затем, – **Enter**.

Команда будет достроена до «cd /usr/local/» и выполнена. Если бы вместо символа «o» был введен символ «i», shell снова не смогла бы достроить имя файла и, после повторного нажатия **Tab**, снова выдала бы списка всех возможных вариантов. Проверьте…?

5. Введите символ «v» и дважды нажмите клавишу **Tab**.

Будет выдан список команд shell, которые начинаются на «v».

6. Введите символ «i» и, снова, дважды нажмите клавишу **Tab**.

Будет выдан гораздо меньший список команд, которые начинаются на «vi».

7. Нажмите клавишу **Enter**.

Будет запущен текстовый редактор «vi».

8. Нажмите сочетание клавиш **Ctrl+Z**.

Текстовый редактор «vi» завершит работу.

9. В пустой командной строке дважды нажмите клавишу **Tab**.

Будет выдано, примерно, такое сообщение: «Display all 2148 possibilities? (y or n)» – «Показать все 2148 возможности? (y или n)».

10. Нажмите клавишу «y».

При помощи клавиши «Пробел» (**Space**) «полистайте» возможные варианты завершения. Для завершения просмотра необходимо, либо «долистать» до конца, либо нажать клавишу «**Q**».

Шаг 4. История команд

Для практического освоения приемов работы с «историей команд» выполните следующее:

1. Несколько раз нажмите клавишу «стрелка вверх» (↑), или сочетание клавиш **Ctrl+P**.

В командной строке будут появляться предыдущие введенные команды.

2. Несколько раз нажмите клавишу «стрелка вниз» (↓), или сочетание клавиш **Ctrl+N**.

В результате будут появляться команды по направлению к последней введенной.

3. Нажмите сочетание клавиш **Ctrl+R**.

В командной строке появится подсказка специального вида – «(reverse-i-search)`': ».

4. Нажмите клавишу «m».

В результате подсказка в командной строке изменится на: «(reverse-i-search)`m': » и появится первая из команд, которая содержит символ «m». Если дальше нажимать **Ctrl+R**, то будут появляться следующие команды, содержащие символ «m».

5. Нажмите клавишу «a».

Подсказка в командной строке снова изменится, и появится первая из команд, которая содержит символы «ma».

6. Самостоятельно, при помощи сочетаний клавиш **Ctrl+R** найдите в истории команд, например, команду «man man».

7. При помощи клавиши **Backspace** очистите содержимое командной строки, или – выполните и завершите найденную команду.

8. В начале командной строки введите символы «ec» и, не нажимая клавишу **Enter**, нажмите клавишу **Tab**.

Получится команда «echo».

9. Дополните ее символами «$HI» и дважды нажмите клавишу **Esc**.

Получится «echo $HIST».

10. Нажимайте клавишу **Esc** до тех пор, пока над командной строкой не появится список возможных вариантов завершения.

11. Нажмите клавишу «**F**».

12. Один раз нажмите клавишу **Tab**, или два раза – клавишу **Esc**.

В результате получится команда «echo $HISTFILE».

13. Нажмите клавишу **Enter**.

Система выдаст полное имя файла, в котором хранится история команд. Например, «/home/student21/.bash_history».

14. Введите команду «history».

В результате будет выдана вся история команд в два столбика: номер команды и сама команда.

15. При помощи сочетаний клавиш **Shift+PgUp** и **Shift+PgDown** «полистайте» последние страницы вывода истории команд.

Шаг 5. Редактирование команд

Для практического освоения приемов редактирования команд выполните следующее:

1. Найдите в истории команду «man man», «man info» или похожую.

2. При помощи клавиш **Home**, **End**, «стрелка влево» (←), «стрелка вправо» (→), **Backspace** и **Delete** отредактируйте строку так, чтобы получилась, например, команда «man ls» и нажмите клавишу **Enter**.

3. Завершите работу команды «man», нажав клавишу «**Q**».

4. Повторите пункты 1 ÷ 3 этого шага, адаптировав их для поиска и редактирования других команд.

5. Найдите в истории, или наберите на клавиатуре, команду, состоящую из 3-х слов (например, «type -at echo»), не нажимая в конце клавишу **Enter**.

6. Нажмите три раза сочетание клавиш **Alt+B**.

Курсор переместится на три слова влево – в начало команды.

7. Нажмите два раза сочетание клавиш **Alt+F**.

Курсор остановится после второго слова.

8. Нажмите сочетание клавиш **Ctrl+W**.

В команде будет удалено второе слово.

9. Введите команду «mroe ~/.bash_history» и нажмите клавишу **Enter**.

Будет выдано диагностическое сообщение о том, что команды «mroe» не существует

10. «Достаньте» команду из истории.

11. Нажимайте «стрелку влево», пока курсор не переместится под символ «o» в слове «mroe».

Можно при помощи удаления и вставки символов отредактировать слово «mroe» необходимым образом – не делайте этого.

12. Нажмите сочетание клавиш **Ctrl+T**.

В результате символы «o» и «r» поменяются местами, и слово «mroe» превратится в «more» – что и требовалось.

13. Можно нажать клавишу **Enter**, не перемещая курсор в конец строки, чтобы запустить команду на выполнение.

Завершить запущенную команду «more», можно, нажав клавишу «**Q**».

Шаг 6. Фильтры

Для практического освоения приемов работы с программами-фильтрами выполните следующее:

1. Введите команду «cd».

Чтобы перейти в домашний каталог.

2. Выполните команду «echo Здравствуй, мир!».

В результате с новой строки будет выдана строка символов «Здравствуй, мир!».

3. Выполните команду «sort».

Курсор остановится в начале пустой строки.

4. Введите следующие слова, после каждого нажимая клавишу **Enter**: «Яблоки», «Груши» и «Сливы».

Все слова должны быть введены в отдельных строчках.

5. Нажмите сочетание клавиш **Ctrl+D**.

Программа-фильтр «sort» закончит работу и выдаст отсортированный результат:

Груши
Сливы
Яблоки.

6. Выполните команду «sort > Фрукты.txt.sort».

7. Повторите пункты 4 и 5 этого шага.

Команда «sort» закончит работу, результат отображаться не будет, а будет записан в файл Фрукты.txt.sort.

8. Выполните команду «cat Фрукты.txt.sort».

Будет выдано содержимое файла Фрукты.txt.sort – отсортированные данные.

9. Удалите файл `Фрукты.txt.sort`.

10. Выполните команду «`cat > Фрукты.txt`».

Будет создан пустой файл «`Фрукты.txt`».

11. Повторите пункты 4 и 5 этого шага.

12. Выполните команду «`cat Фрукты.txt`».

Будет выдано содержимое файла `Фрукты.txt`.

13. Выполните команду «`sort Фрукты.txt`».

Программа «`sort`» в качестве входных данных читает файл `Фрукты.txt`, а выдает – отсортированный результат.

14. Выполните команду «`sort Фрукты.txt > Фрукты.txt.sort`».

Результат сортировки будет записан в файл `Фрукты.txt.sort`.

15. Посмотрите файл `Фрукты.txt.sort`.

Он должен быть отсортированным.

16. Выполните команду «`wc -w`».

Курсор остановится в начале пустой строки.

17. Введите следующий текст: «`Здравствуй, мир!`», и нажмите клавишу **`Enter`**.

18. Нажмите сочетание клавиш **`Ctrl+D`**.

Чтобы завершить ввод с терминала. Будет выдано число 2 – количество слов, которое содержит фраза «`Здравствуй, мир!`».

19. Выполните команду «`wc Фрукты.txt`».

Будет выведено количество строк, слов и байт в файле `Фрукты.txt`.

20. Выполните команду «`wc -l Фрукты.txt.sort`».

Будет выведено количество строк в файле `Фрукты.txt.sort`.

21. Самостоятельно, определите при помощи одной команды количество слов и байт в файле `Фрукты.txt.sort`.

22. Удалите файлы `Фрукты.txt` и `Фрукты.txt.sort`.

Для этого введите следующие символы «`rm Фру`» и, не нажимая клавишу **`Enter`**, нажмите клавишу **`Tab`** – команда будет достроена до: «`rm Фрукты.txt`». Введите символ «*» и нажмите клавишу **`Enter`**. Далее, подтвердите удаление файлов – нажимая клавишу «**`y`**» («yes»).

Шаг 7. Перенаправление ввода-вывода

Для практического освоения приемов перенаправления ввода-вывода выполните следующее:

1. Введите команду «`cd`».

Чтобы перейти в домашний каталог.

2. Введите команду «`ls -l /usr/bin`».

Содержимое каталога «`/usr/bin`» быстро промелькнет по экрану.

3. Введите команду «`ls -l /usr/bin > dir.txt`».

Перечень файлов, находящихся в каталоге `/usr/bin`, будет записан в файл `dir.txt`.

4. Просмотрите содержимое файла `dir.txt`.

Это будет перечень файлов, находящихся в каталоге `/usr/bin`.

5. Введите команду «`ls -l /usr/sbin > dir.txt ; less dir.txt`».

В начале, перечень файлов, находящихся в каталоге `/usr/sbin`, будет записан в файл `dir.txt` а, затем, файл `dir.txt` – просмотрен с помощью команды «`less`». Теперь это будет перечень файлов, находящихся в каталоге `/usr/sbin` – прежнее содержание файла было уничтожено. Символ «`;`» служит для разделения двух команд в одной командной строке.

6. Введите команду «`date >> dir.txt ; less dir.txt`».

В файле `dir.txt` будет перечень файлов, находящихся в каталоге `/usr/sbin`, а в конце – текущая дата.

7. Удалите файл `dir.txt`.

8. Введите команду «`ls -l /usr/bin | less`».

При помощи средств управления команды «`less`» «полистайте» содержимое каталога `/usr/bin`. Для завершения команды «`less`» нажмите клавишу «**Q**».

9. Введите команду «`ls -l /usr/bin | wc -l`».

Будет выдано число, на единицу больше количества файлов в каталоге `/usr/bin`. Команда «`wc`» с ключом «`-l`» подсчитывает число строк в файле – в канале, куда направлен вывод команды «`ls`».

10. Введите команду «`echo Файлов: `ls -l /usr/bin | wc -l`».

Символ «одинарной обратной кавычки» («`») находится на одной клавише с символом «тильда» («~»). Вывод команды «`echo`» – слово «`Файлов:`» – будет дополнен числом, на единицу большем количества файлов в каталоге `/usr/bin` – выводом команды, заключенной в одинарные обратные кавычки – «`ls -l /usr/bin | wc -l`». В большинстве случаев такие «длинными» командами используются довольно редко. Для этого в shell имеется специальный механизм – «*сценарии*» – специальные файлы, в которые записываются последовательности «старых» команд, и получаются «новые», более короткие, команды.

11. Введите команду «`ls -l /usr/bin | tee dir.txt | less`».

Выход команды «`ls`» – перечень файлов каталога `/usr/bin` – передается но вход команды «`tee`», которая записывает его в файл `dir.txt`, а копию передает на вход следующей команды – «`less`». При помощи команды «`less`» можно спокойно «полистать» список файлов, которые находятся в каталоге `/usr/bin`. Обратите внимание на количество строк (файлов) вывода. В результате при помощи одной команды был просмотрен каталог `/usr/bin`, а копия вывода – записана в файл `dir.txt`.

12. Посмотрите содержимое файла `dir.txt`.

Оно должно соответствовать выводу в предыдущем пункте. Обратите внимание на количество строк (файлов) вывода.

13. Удалите файл `dir.txt`.

Шаг 8. Сценарии

Для практического освоения приемов работы со сценариями (командными файлами) выполните следующее:

1. Введите команду «`echo Здравствуй, Мир!`».

Будет выдано сообщение «`Здравствуй, Мир!`».

2. Введите команду «`cat > Hello.script`».
3. Введите строку «`echo Здравствуй, Мир!`».

Для завершения ввода нажмите **`Enter`**, а затем – **`Ctrl+D`**.

4. Введите команду «`Hello.script`».

Будет выдано диагностическое сообщение о том, что команда `Hello.script` не найдена.

5. Введите команду «`./Hello.script`».

Чтобы явно указать, где искать сценарий `Hello.script` – в текущем каталоге – «`./`». В результате будет выдано диагностическое сообщение о том, что отказано в доступе – файл `Hello.script` не является исполняемым.

6. Введите команду «`sh Hello.script`».

Для запуска скрипта необходимо запустить еще один экземпляр shell и передать ей в качестве параметра имя файла сценария. Попробуйте запустить сценарий в другой оболочке, например, «`bash Hello.script`». Если скрипт выдает не то, что надо – повторите пункты 2, 3 и 6 текущего шага.

7. Введите команду «`ls -l Hello.script`».

Посмотрите параметры доступа к файлу `Hello.script`. Обратите внимание, что по умолчанию файл создается доступным на запись и чтение, но не на выполнение.

8. Введите команду «`chmod +x Hello.script`».

Для всех категорий пользователей будет установлен параметр доступа к файлу `Hello.script` «исполняемый».

9. Повторите команду «`ls -l Hello.script`».

Посмотрите параметры доступа к файлу `Hello.script`.

10. Введите команду «`Hello.script`».

Будет выдано диагностическое сообщение о том, что команда `Hello.script` не найдена.

11. Введите команду «`./Hello.script`».

Теперь скрипт сработает, поскольку явно указано, где его искать – в текущем каталоге – «`./`». Для того, чтобы скрипт можно было запускать, используя только имя файла, его необходимо поместить в один из каталогов, перечисленных в переменной окружения `PATH`. Например, в каталог `/home/student21/bin` – только если он есть в списке путей поиска команд.

12. Введите команду «`mv Hello.script bin`».

Скрипт `Hello.script` будет перемещен в существующий каталог `/home/student21/bin` – каталог личных команд (программ) пользователя `student21`.

13. Введите команду «ls Hello.script».

Будет выдано сообщение, которое говорит о том, что в текущем каталоге файла Hello.script нет.

14. Посмотрите содержимое каталога/home/student21/bin.

Там должен быть файл Hello.script.

15. Введите команду «Hello.script».

Теперь все сработало как надо.

16. Перейдите в каталог/home/student21/bin.

17. Введите команду «cat > sort.script».

18. Введите следующие фрагменты текста, после каждого, нажимая клавишу **Enter**: «#!/bin/sort», «Яблоки», «Груши» и «Сливы».

Все фрагменты текста должны быть введены в отдельных строчках.

19. Нажмите сочетание клавиш **Ctrl+D**.

Чтобы закончить создание сценария sort.script.

20. Посмотрите содержание файла sort.script.

21. Установите для файла sort.script параметр доступа «исполняемый».

22. Введите команду «sort.script».

Будет выдано отсортированное содержание самого скрипта. Если результат будет иным – повторите пункты этого шага, начиная с 16-го.

23. Создайте сценарий nf такого содержания:

```
#!/bin/sh
# nf - выводит кол-во файлов в каталоге
echo Сегодня: `date`
echo в каталоге $* `ls $* | wc -l` файлов
```

Он выводит общее количество файлов в каталогах, имена которых переданы ему в качестве параметров.

24. Проверьте работу скрипта nf без параметров (текущий каталог), с одним параметром (например, родительский каталог) и с несколькими параметрами (например, текущий и родительский каталоги).

25. Удалите созданные ранее сценарии: Hello.script, sort.script и nf.

Индивидуальные задания

1. .Посмотреть содержание файла истории команд.
2. Определить максимальное количество команд, которые может содержать файл истории.
3. При помощи конвейера из команды file и less просмотрите каталоги /bin и /usr/bin, чтобы узнать, какие из команд являются сценариями.
4. В тетрадь для выполнения лабораторных работ занести полученные результаты, и как они были получены.
5. Показать работу преподавателю.
6. Удалить, созданные в ходе лабораторной работы, файлы и каталоги.

Контрольные вопросы

1. Что такое shell? Назначение. Назовите синонимы.
2. Назовите основные группы Shell. Чем они отличаются?
3. Как определить название текущей shell?
4. Приведите структуру команды ОС Linux.
5. На какие группы делятся команды ОС Linux?
6. Как определить тип команды?
7. Что делает команда `echo`? К какому типу она относится?
8. Как посмотреть каталоги поиска утилит ОС Linux?
9. Как восстановить сбившиеся настройки терминала в исходное состояние?
10. Как в ОС Linux выполняется «достраивание» команд?
11. Какие есть варианты «достраивания» команд, если с первой попытки не получилось?
12. Что такое «история команд»? Зачем она нужна? Как ею пользоваться?
13. Как посмотреть максимальное количество команд, которое может храниться в «истории»?
14. Как отыскать нужную команду в «истории», не пролистывая ее всю, если известно только несколько ее символов?
15. Как посмотреть имя файла «истории», в котором сохраняются команды между сеансами работы? Где он обычно находится?
16. Как посмотреть максимальное количество команд, которое может храниться в файле «истории»?
17. Что такое стандартные устройства ввода и вывода в ОС Linux? Как они называются?
18. Как перенаправить вывод команды ОС Linux в файл?
19. Как дописать вывод команды ОС Linux в уже существующий файл??
20. Как перенаправить вывод одной команды ОС Linux на вход другой? Приведите пример.
21. Для чего служит символ «`;`» в командной строке?
22. Как поместить вывод команды ОС Linux в командную строку?
23. Что такое программы-фильты? Зачем они нужны? Как используются?
24. Приведите примеры программ-фильтров. Что ни делают?
25. Как одновременно записать вывод команды в файл и вывести на терминал?
26. Что такое сценарии? Зачем они нужны? Назовите синонимы.
27. Что необходимо сделать, чтобы сценарий можно было запускать на выполнение, введя только его имя?
28. Назовите характерный признак «правильного» сценария?
29. Как написать сценарий, который будет выполнять нужная программа-утилита?
30. Как получить исчерпывающую информацию по работе shell?
31. Сформулируйте и задайте свой вопрос по пройденному материалу.
32. Сформулируйте и внесите свое предложение по совершенствованию темы.

Лабораторная работа № 4
Создание и редактирование Web-страниц

Цель работы

Освоение и приобретение практических навыков работы с инструментарием для создания Web-страниц, их создание и редактирование, управление цветом текста и фона.

Краткие теоретические сведения

Для создания Web-страниц используются различные программные средства, продуктивная и комфортная работа с которыми в немалой степени зависит от их оптимальной совместной настройки. В простейшем случае текст документа HTML можно подготовить при помощи стандартного текстового редактора, который по умолчанию имеется в любой операционной системе. Например, в Windows – это Блокнот (Notepad), а в Linux – Mousepad. Посмотреть же созданную Web-страницу можно в программе-браузере, например, Firefox – для Linux или Internet Explorer – для Windows. Поскольку текстовые редакторы изначально предназначены для подготовки текстовых документов, то по умолчанию они при сохранении документа либо вообще не ставят ни какого расширения имени файла (например, Mousepad в Linux), либо используют расширение `.txt` (Блокнот в Windows). Поэтому при сохранении Web-страниц необходимо явно указывать расширение имени файла `.htm` или `.html`. Для того, что бы можно было увидеть отредактированный HTML-документ, необходимо не забывать сохранять его в программе текстового редактора и обновлять содержимое окна браузера.

Однако, иногда при просмотре Web-страницы в окне браузера отображается не читабельный текст, а «кракозябры». Так происходит потому, что некоторые браузеры автоматически не правильно определяют кодировку текста HTML-документа. Для корректного отображения Web-страницы в браузере необходимо выполнить команду **Вид ⇨ Кодировка** и указать кодировку, в которой она была подготовлена. Чтобы всякий раз при несовпадении кодировок в текстовом редакторе и браузере не выполнять эту команду необходимо один раз установить в нем предпочитаемую кодировку.

Аналогичная ситуация иногда наблюдается также при переносе HTML-документов из Windows в Linux или наоборот – из Linux в Windows. По умолчанию текст в операционной системе Linux кодируется по таблице символов **Юникод (UTF-8)**, а в Windows – **Кириллица (Windows-1251)**. Поэтому для обеспечения единообразия необходимо сохранять текст HTML-документов и в Linux и в Windows в одной кодировке. Например, при сохранении HTML-документа в программе Блокнот явно указывать кодировку – **UTF-8**.

Практически все современные браузеры по умолчанию настроены на использование своей внутренней программы просмотра кода Web-страниц, которая позволяет только просматривать, но не редактировать HTML-код. Однако, в большинстве случаев всегда можно настроит браузер так, чтобы для

просмотра и редактирования HTML-кода страницы он запускал необходимую внешнюю программу.

Любой документ HTML состоит из текста, который необходимо отобразить, и инструкций, которые управляют способом отображения. Такие инструкции называются ***тегами***. Чтобы браузер мог отличить текст, который необходимо отобразить, от тегов последние заключаются в символы «<» «>». Текст HTML-документа должен начинаться тегом `<html>`, а заканчиваться `</html>`. Внутри этих тегов помещается «*голова*» и «*тело*» документа, которые обозначаются тегами `<head>...</head>` и `<body>...</body>`, соответственна. Тег `<title>...</title>`, помещаемый внутри тега `<head>...</head>`, служит для указания информации, которая будет отображаться в заголовке окна браузера. Информация, которая должна отображаться в основном окне браузера, помещается внутри тега `<body>...</body>`. Большинство браузеров не чувствительны к регистру символов в тегах, поэтому их можно записывать как строчными, так и прописными буквами, или их сочетанием. Например, правильно будут обработаны такие теги, как `<BODY>`, `<body>` и `<Body>`.

Цвет фона и символов HTML-документа можно указать двумя способами:

1. в формате ***RGB***, когда он указывается долями трех его составляющих:
   а) **красной (R)** – от англ. **Red**,
   б) **зеленой (G)** – от англ. **Green** и
   в) **синей (B)** – от англ.**Blue** или
2. его английским названием.

При указании цвета в формате ***RGB*** он получается в результате сложения трех его компонентов. Интенсивность каждой из этих составляющих указывается своим 16-ричным значением в диапазоне от `00` до `FF` – чем больше число, тем интенсивнее этот цвет в результирующем. Примеры записи цветов приведены в Таблице 1.

Таблица 1

Название цвета			**Название цвета**		
Русское	**Английское**	**RGB**	**Русское**	**Английское**	**RGB**
черный	`black`	`000000`	фиолетовый	`purple`	`FF00FF`
белый	`white`	`FFFFFF`	желтый	`yellow`	`FFFF00`
красный	`red`	`FF0000`	коричневый	`brown`	`996633`
зеленый	`green`	`00FF00`	оранжевый	`orange`	`FF8000`
бирюзовый	`azure`	`00FFFF`	лиловый	`violet`	`8000FF`
синий	`blue`	`0000FF`	серый	`gray`	`A0A0A0`

Цвет фона устанавливается при помощи атрибута `bgcolor=` в теге `<body>`. Если он не указан, то по умолчанию используется цвет, указанный в настройках браузера. Как правило – это белый цвет.

Цвет текста, отображаемого в окне браузера по умолчанию, – черный. Он указывается в его настройках. Явно цвет текста можно указать в теге `<body>`

при помощи атрибута `text=`. Цвет отдельного фрагмента текста изменяется при помощи атрибута `color=` тега `<font>`.

Все, с теорией закончили. Настала пора практики. В конце концов, учиться лучше всего на примерах. И дальше – по инструкции:

Ход работы

Шаг 1. Настройка инструментария и создание шаблона Web-страницы

Чтобы создать HTML-документ, выполните следующее:

1. В своем рабочем каталоге, например:
   - `/home/student1/ivanov/` (для Linux) или
   - `D:\course1\БУА-1\Иванов\` (для Windows) или
   - указанном преподавателем

   создайте каталог `html`, в котором будут храниться документы HTML, создаваемые по ходу выполнения лабораторных работ.
2. В созданном ранее каталоге `html` создайте каталог `lr1` и сделайте его текущим.

   В каталоге `lr1` будут храниться документы HTML, создаваемые по ходу выполнения текущей лабораторной работы.
3. В каталоге `lr1` на свободном месте щелкните правой кнопкой мыши и в раскрывшемся контекстном меню выполните команду:
   - **Создать документ ⇨ Пустой файл** (для Linux) или
   - **Создать ⇨ Текстовый документ** (**Plain Text**) (для Windows)

   и в раскрывшемся окне диалога укажите в качестве имени файла `template.html`.

   В случае выдачи запроса на изменение расширения имени файла – подтвердите.
4. Щелкните правой кнопкой мыши по вновь созданному файлу `template.html` и в раскрывшемся контекстном меню выполните команду:
   - **Открыть с помощью "Текстовый редактор Mousepad"** или
   - **Открыть с помощью ⇨ Открыть с помощью "Текстовый редактор Mousepad"** (для Linux) или
   - **Открыть с помощью ⇨ Блокнот** (**Notepad**) (для Windows).

   Если указанные редакторы отсутствуют в контекстном меню, то их необходимо найти самостоятельно при помощи последнего пункта:
   - **Открыть с помощью другого приложения…** (для Linux) или
   - **Выбрать программу…** (для Windows).
5. В раскрывшемся окне редактора наберите следующий код (текст):

```
<html>
<head>
<title></title>
</head>
<body>
</body>
</html>
```

6. Сохраните файл, выполнив команду **Файл ⇨ Сохранить**.

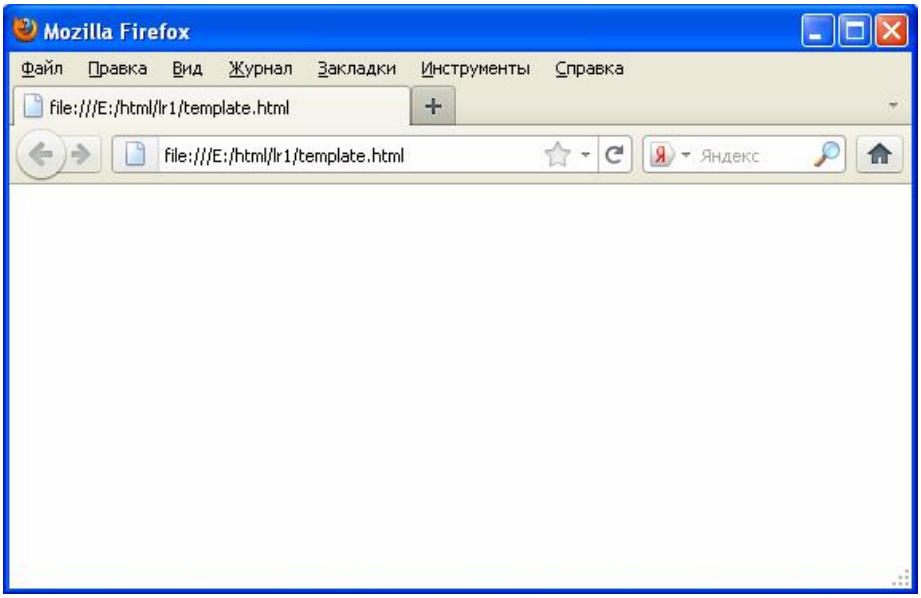
7. Для просмотра созданной Web-страницы щелкните правой кнопкой мыши по файлу `template.html` и в раскрывшемся контекстном меню выполните команду:

- **Открыть с помощью "Интернет-проводник Firefox"** или
- **Открыть с помощью ⇨ Открыть с помощью "Интернет-проводник Firefox"** (для Linux) или
- **Открыть с помощью ⇨ Internet Explorer** (для Windows) или
- выберете любой другой браузер, установленный в Вашей системе.

Если необходимый браузер отсутствует в меню, то его необходимо найти самостоятельно при помощи последнего пункта:

- **Открыть с помощью другого приложения…** (для Linux) или
- **Выбрать программу…** (для Windows).

В результате откроется пустое окно браузера:



Так происходит потому, что созданный файл не содержит никакой полезной информации, а является всего лишь заготовкой (шаблоном) для любого HTML-документа, т. е. в дальнейшем его можно будет брать за основу при создании произвольной Web-страницы.

8. В текстовом редакторе отредактируйте файл `template.html` так, чтобы между тегами `<title>` и `</title>` был помещен текст «`Мой первый документ HTML`», а между `<body>` и `</body>` – «`Всем огромный привет!`».

В результате документ HTML должен иметь такое содержание:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body>
Всем огромный привет!
</body>
</html>
```

9. Сохраните отредактированный документ в файле `hello.html`, выполнив команду **Файл ⇨ Сохранить как…**.

10. Закройте браузер.

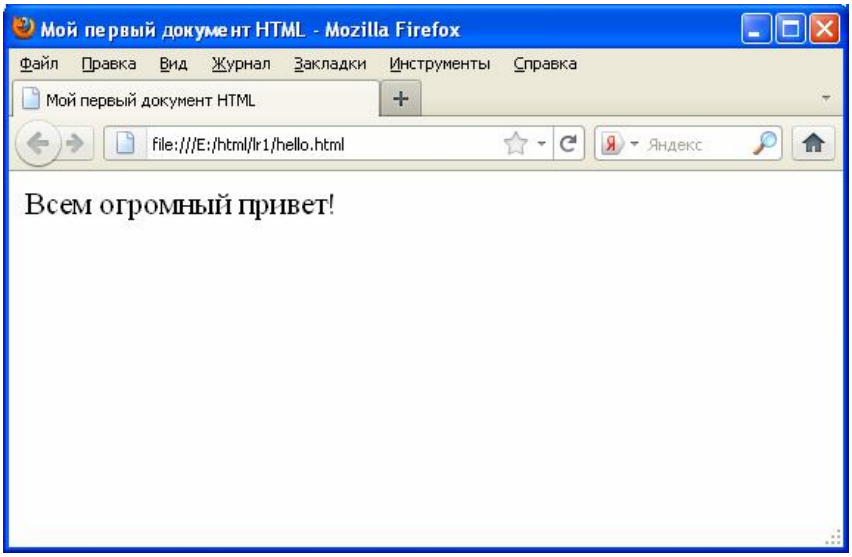
11. Щелкните правой кнопкой мыши по вновь созданному файлу `hello.html` и в раскрывшемся контекстном меню выберете команду **Свойства**.

12. В раскрывшемся окне диалога свойств файла `hello.html`:

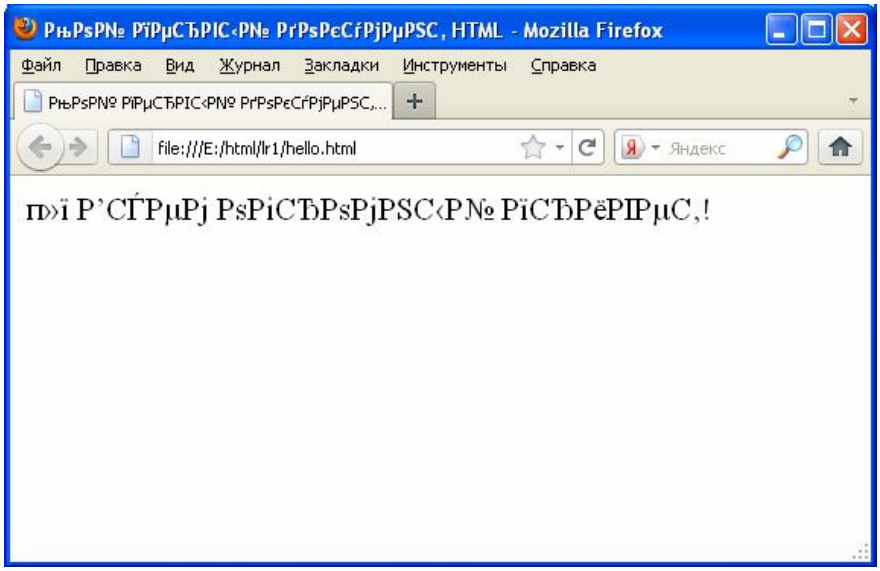
- на вкладке **Основные** при помощи раскрывающегося списка **Открывать с помощью:** укажите, например, **Интернет-проводник Firefox** (для Linux) или
- на вкладке **Общие** в групповой рамке **Тип файла: Приложение:** щелкните по кнопке **Изменить** и в раскрывшемся далее окне диалога **Выбор программы** укажите, например, **Internet Explorer** (для Windows).

13. Выполните двойной щелчок по файлу `hello.html`.

В результате будет запущен, выбранный в предыдущем пункте, браузер и Вы увидите примерно следующее:



Если же в окне браузера вместо читабельного текста будут отображаться «кракозябры» как, например:



то это лишь означает, что браузер не смог автоматически определить кодировку русского (кириллического) текста. Для правильного отображения текста необходимо в браузере выполнить команду **Вид ⇨ Кодировка** и выбрать:

- **Юникод (UTF-8)** – если текст Web-страницы был набран в Linux или
- **Кириллица (Windows-1251)** – если текст Web-страницы был набран в Windows.

Такая «ошибка» может возникнуть также в ситуации когда, например, текст Web-страницы был подготовлен в операционной системе Windows, а просматривается браузером в Linux или, наоборот. По умолчанию текст в операционной системе Linux кодируется в кодировке **Юникод (UTF-8)**, а в Windows – **Кириллица (Windows-1251)**. Для обеспечения единообразия в такой ситуации рекомендуется при сохранении файла программой Блокнот (Notepad) в операционной системе Windows в окне диалога **Сохранить как** в раскрывающемся списке **Кодировка** указать значение **UTF-8**.

При редактировании текста, набранного в Windows, без перекодировки в текстовом редакторе Mousepad операционной системы Linux необходимо при его открытии правильно указать исходную кодировку. Для этого в переключателе кодировок установить значение **Иная**, а в раскрывающемся списке – **WINDOWS-1251**.

Чтобы всякий раз при несовпадении кодировок в текстовом редакторе и браузере не выполнять команду **Вид ⇨ Кодировка** необходимо установить в нем предпочитаемую кодировку. Для этого, например, в браузере Firefox в окне диалога **Настройки** на вкладке **Содержимое** в групповой рамке **Шрифты и цвета** щелкните по кнопке **Дополнительно…** и в раскрывшемся окне диалога **Шрифты** при помощи раскрывающегося списка **Кодировка по умолчанию** выставьте необходимое значение, например, **Юникод (UTF-8)**. Также правильному распознаванию кодировки русского текста способствует правильная установка значения **Русская** (или, наоборот, **(Отключено)**) по команде **Вид ⇨ Кодировка ⇨ Автоопределение**.

Шаг 2. Настройка редактора и задание цвета текста

Чтобы установить цвет текста …, выполните следующее:

1. Закройте программу текстового редактора.
2. В окне браузера на свободном месте щелкните правой кнопкой мыши и в раскрывшемся контекстном меню выберите пункт **Исходный код станицы** (или **Просмотр HTML кода** или **Просмотр кода страницы**) или в основном меню **Вид** (или **Инструменты ⇨ Веб-разработка**) выберете аналогичный пункт.

В результате в большинстве современных браузеров откроется дополнительное окно, в котором будет исходный HTML-код страницы, редактировать который нельзя – его можно только просматривать.

Поэтому, … есть два варианта:

- настроить браузер так, чтобы для просмотра и редактирования кода Web-страницы он запускал необходимый внешний редактор (далее, пункт 4), или

• для редактирования Web-страницы запускать внешний редактор через файловый менеджер (Проводник, Thunar и т. д.) по щелчку на нем правой кнопки мыши (далее, пункт 5).

3. Закройте окно с HTML-кодом.

4. Настройте браузер так, чтобы для просмотра и редактирования кода Web-страницы он запускал необходимый внешний редактор.

Для этого:

• *в Internet Explorer*: выполните команду **Сервис ⇨ Средства разработчика** (или **F12**), и в раскрывшемся окне выполните команду **Файл ⇨ Настроить источник представлений Internet Explorer**. А далее, выберите **Блокнот** – чтобы править код в Блокноте, или **Прочие …** – чтобы подключить любой другой внешний редактор.

• *в Firefox*: перейдите по адресу `about:config`; при этом откроется окно диалога с предупреждением об осторожном изменении настроек. Затем необходимо установить значение параметра `view_source.editor.external` в `true`, а в `view_source.editor.path` – записать путь к требуемому редактору. Например, C:\WINDOWS\system32\notepad.exe для программы Блокнот в Windows, или – `/usr/bin/mousepad` – для Mousepad в Linux.

Теперь для редактирования Web-страниц по щелчку правой кнопкой мыши в окне браузера можно будет запускать выбранный редактор.

Или – второй вариант:

5. Выполните щелчок правой кнопкой мыши по файлу `hello.html`, а в раскрывшемся контекстном меню выберите пункт **Открыть с помощью** и укажите необходимый текстовый редактор.

6. Измените цвет слова «Всем» на красный, указав название цвета.

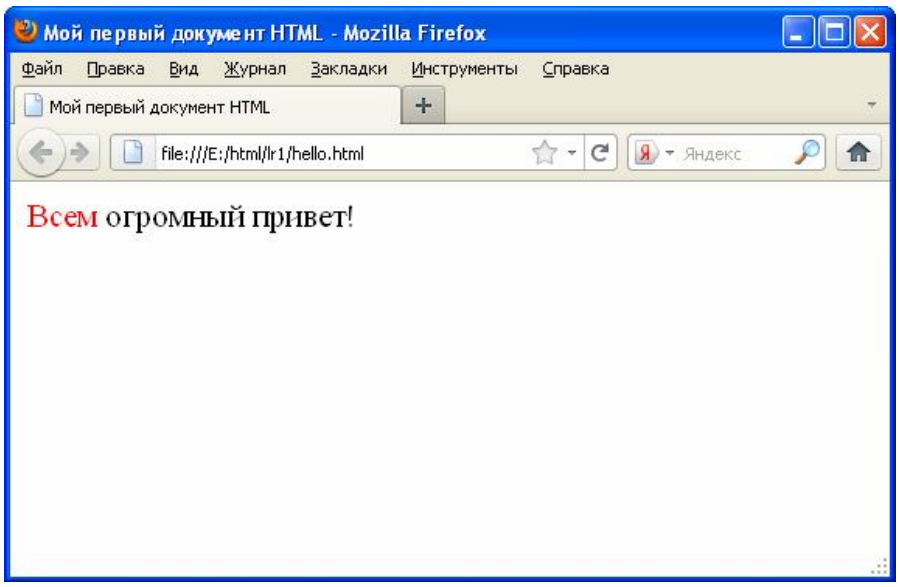
Для этого файл `hello.html` отредактируйте следующим образом:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body>
<font color="red">Всем</font> огромный привет!
</body>
</html>
```

7. Сохраните внесенные в файл `hello.html` изменения, выполнив команду **Файл ⇨ Сохранить**.

8. Обновите содержимое окна браузера нажав, соответственно, кнопку **Обновить** (**Обновить текущую страницу, Обновить эту страницу**) или клавишу **F5**.

В результате его окно должно выглядеть примерно так:



Если Web-страница сильно отличается от представленной, то повторите три последних пункта еще раз.

9. Отредактируйте файл `hello.html` так, чтобы цвет слова «**огромный**» был **зеленым**, указав цвет в формате ***RGB***:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body>
<font color="red">Всем</font> <font
color="#00FF00">огромный</font> привет!
</body>
</html>
```

Обратите внимание, что значению цвета обязательно должен предшествовать знак «решетка» – «#».

10. Просмотрите отредактированную Web-страницу.

Для этого:

а) сохраните ее в текстовом редакторе и

б) обновите содержание окна браузера.

11. Установите весь остальной текст на странице синим.

Для этого отредактируйте файл `hello.html` следующим образом:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body text="#0000FF">
<font color="red">Всем</font> <font
color="#00FF00">огромный</font> привет!
</body>
</html>
```

12. Просмотрите в браузере отредактированную Web-страницу.

Шаг 3. Задание цвета фона

Чтобы установить цвет фона выполните следующее:

1. Отредактируйте файл `hello.html` следующим образом и просмотрите его в браузере:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body text="#0000FF" bgcolor="#000000">
<font color="red">Всем</font> <font
color="#00FF00">огромный</font> привет!
</body>
</html>
```

Индивидуальные задания

1. В созданном ранее каталоге `lr1` создайте каталог `iz`.

В нем будут храниться документы HTML, создаваемые по ходу выполнения индивидуального задания текущей лабораторной работы.

2. В соответствии с номером по списку группы, или указанием преподавателя, установите такие цвета на Web-странице, а так же обратите внимание на их сочетаемость и уместность:

Вариант	Фон	Текст	Слово «Всем»	Слово «огромный»
1	белый	черный	серый	синий
2	черный	белый	красный	серый
3	черный	зеленый	серый	красный
4	синий	белый	зеленый	бирюзовый
5	лиловый	белый	желтый	оранжевый
6	серый	зеленый	лиловый	бирюзовый
7	красный	фиолетовый	лиловый	зеленый
8	фиолетовый	зеленый	серый	синий
9	зеленый	синий	красный	серый
10	серый	черный	зеленый	лиловый
11	лиловый	зеленый	фиолетовый	черный
12	коричневый	зеленый	серый	красный
13	синий	серый	зеленый	черный
14	черный	зеленый	серый	лиловый
15	оранжевый	лиловый	фиолетовый	белый
16	зеленый	синий	оранжевый	серый
17	лиловый	зеленый	черный	синий
18	бирюзовый	черный	белый	коричневый
19	оранжевый	фиолетовый	лиловый	серый
20	серый	зеленый	оранжевый	коричневый
21	коричневый	белый	зеленый	красный
22	красный	зеленый	черный	коричневый
23	белый	оранжевый	зеленый	фиолетовый
24	фиолетовый	бирюзовый	лиловый	черный
25	черный	фиолетовый	оранжевый	серый
26	серый	оранжевый	фиолетовый	красный
27	оранжевый	фиолетовый	красный	серый
28	желтый	лиловый	фиолетовый	бирюзовый
29	бирюзовый	красный	коричневый	лиловый
30	зеленый	синий	оранжевый	черный

3. Созданную Web-страницу сохраните в каталоге `iz` под именем `my.html`.

4. В тетрадь для выполнения лабораторных работ занесите эскиз созданной Web-страницы и, соответствующей ей, текст (код) HTML-документа.

Контрольные вопросы

1. С помощью какой программы, в простейшем случае, можно подготовить HTML-документ в Windows и Linux, соответственно?
2. Как в программе текстового редактора сохранить созданный HTML-документ?
3. С помощью какой программы можно просмотреть созданную Web-страницу?
4. Почему в браузере могут не отображаться изменения HTML-документа, выполненные в программе текстового редактора?
5. Почему иногда браузер вместо читабельного текста отображает «кракозябры»?
6. Как настроить браузер, чтобы он правильно отображал текст Web-страницы?
7. В какой кодировке по умолчанию кодируется текст в операционной системе Windows, а в какой – в Linux?
8. Как необходимо переносить файлы HTML-документов из Windows в Linux, и наоборот, чтобы они правильно отображались в браузере?
9. Как необходимо настроить файловый менеджер, чтобы при двойном щелчке по файлу HTML-документа он открывался в нужной программе-браузере?
10. Как необходимо настроить браузер, чтобы при просмотре исходного HTML-кода страницы его можно было редактировать?
11. Для чего предназначен язык HTML?
12. Для чего служат теги?
13. Как браузер отличает теги от остального текста документа?
14. Из каких обязательных частей состоит документ HTML?
15. Какие теги обязательно присутствуют в любом документе HTML?
16. Как установить заголовок окна браузера?
17. Где записывается информация, которая должна отображаться в основном окне браузера?
18. Для чего служит тег <html>?
19. Для чего служит тег <head>?
20. Для чего служит тег <title>?
21. Для чего служит тег <body>?
22. Какие атрибуты может иметь тег <body>?
23. Как браузеры отображают текст и теги в различных регистрах?
24. Сколько есть способов указания цвета фона и текста HTML-страницы?
25. Как указать цвет в формате ***RGB***?
26. Как установить цвет отдельного фрагмента текста на Web-странице?
27. Как установить цвет всего текста на Web-странице?
28. Какой цвет текста на Web-странице используется по умолчанию?
29. Как установить цвет фона страницы HTML?
30. Какой цвет фона на Web-странице используется по умолчанию?
31. Сформулируйте и задайте свой вопрос по пройденному материалу.
32. Сформулируйте и внесите свое предложение по совершенствованию темы.

Лабораторная работа № 5
Форматирование Web-страниц

Цель работы

Освоение и приобретение практических навыков форматирования Web-страниц, строк и символов.

Краткие теоретические сведения

При обработке HTML-документов браузеры автоматически размещают текст на экране, не принимая во внимание встречающиеся в файле переводы строк и идущие подряд пробелы. Для «разлома» строк служит тег перевода строки `<br>` – он выполняет переход на новую строку. Тэг абзаца `<p>` тоже отделяет строку, но еще добавляет пустую строку, которая зрительно выделяет абзаца.

Тэги выделения фрагментов текста позволяют управлять отображением отдельных символов и слов. Существует три тэга выделения фрагментов текста:

1. `<b>…</b>` – для выделения **полужирным**,
2. `<i>…</i>` – для выделения *курсивом,*
3. `<u>…</u>` – для выделения подчеркиванием.

Большинство тегов, используемых для разметки HTML-страниц, являются контейнерами, т. е. внутри себя они могут содержать текст и другие теги. Например, использование комбинированных начертаний шрифтов:
`<i><b>Всем</b></i> <i>огромный</i> <u>привет!</u>`.

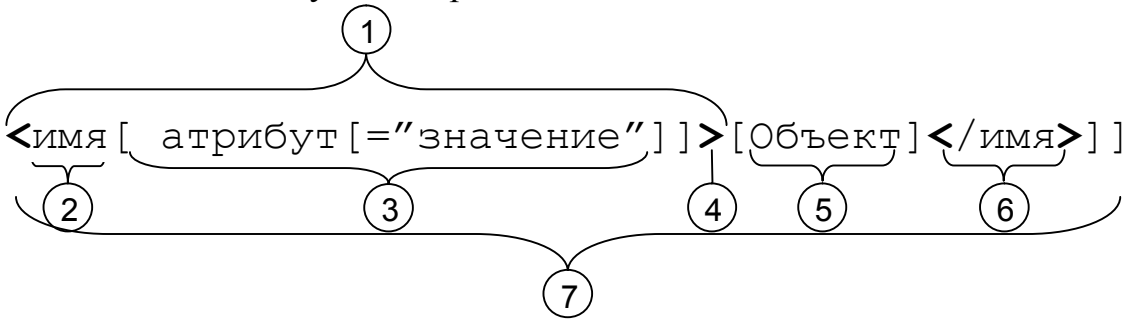
При этом необходимо помнить следующее правило записи вложенных тэгов: тег, который был открыт первым, закрывается последним, вторым – предпоследним и т. д. Говоря иначе, теги со всем их содержимым должны полностью вкладываться в другие теги, не оставляя «хвостов» снаружи. Например,

`<Тэг1> <Тэг2> <Тэг3> … </Тэг3> </Тэг2> </Тэг1>` правильная запись

`<Тэг1> <Тэг2> <Тэг3> … </Тэг3> <Тэг1> </Тэг2>` ошибочная запись

Тег, в который непосредственно вложен данный тег, называется *родительским*, или *родителем*. В свою очередь, тег, вложенный в данный тег, называется *дочерним*, или *потомком*. Уровень вложенности тегов в HTML-коде обозначают с помощью отступов, которые ставят слева от соответствующего тега и создают с помощью пробелов. На отображение Web-страницы они никак не влияют.

Теги вместе с объектами (фрагментами документа), отображением которых он управляет, называются *элементами*. Формальный синтаксис элемента HTML-документа представлен на Рис. 1.



1. *Открывающий тег*
2. *Имя тега*
3. *Атрибут*
4. *Символ-ограничитель*
5. *Отображаемый объект*
6. *Закрывающий тег*
7. *Элемент*

Рис. 1. Элемент HTML-документа

Составляющие в квадратных скобках «[]» являются не обязательными.

Имя тега указывает на характер выполняемых им действий по разметке и форматированию, а атрибуты (параметры) уточняют способ выполнения этих действий. Атрибуты являются не обязательными составляющими тега и если они имеются, то отделяются от имени, и друг от друга, по крайней мере, одним пробелом. Порядок следования атрибутов тега произволен.

Существует два способа управления размером текста, отображаемого браузером:

1. использование стилей заголовка,
2. задание размера текущего шрифта.

В языке HTML имеется шесть тэгов заголовков (от `<h1>` до `<h6>`). Каждому такому тэгу соответствует свой стиль, заданный в параметрах настройки браузера. Из них `<h1>` – самый крупный.

Атрибут `size=` тега `<font>` позволяет задавать размер текущего шрифта для отдельных фрагментов текста. Диапазон установки размера текущего шрифта – от 1 до 7.

Тэг `<font>` многофункциональный – он предоставляет возможности управления размером, начертанием и цветом текста. Изменение гарнитуры шрифта выполняется простым добавлением к тэгу `<font>` атрибута `face=`. Например, для отображения текста шрифтом Arial необходимо указать `<font face="arial">`. Для изменения цвета шрифта в тэге `<font>` используется атрибут `color=`.

С помощью тэгов языка HTML можно управлять горизонтальным выравниванием текста. Если не оговаривать способ выравнивания, то все элементы в документе будут выравниваться по левому краю и иметь неровное правое поле. Для выравнивания текста используется атрибут `align=`, который встраивается в теги абзаца `<p>` или заголовка `<h>` и может принимать такие значения:

1. `center` – выравнивание по центру,
2. `right` – выравнивание по правому краю,
3. `left` – выравнивание по левому краю.

Иногда на Web-странице необходимо отобразить символы, которых нет на клавиатуре, например, «знак зарегистрированной торговой марки» – (®) или символы, которые имеют специальное значение, например, «<» и «>» – начало и окончание тега. В этих случаях можно вводить нужные символы с помощью специальных кодов. Такие коды начинаются с символа «амперсанд» (`&`). За ним следует название символа либо его числовой код в десятичной или шестнадцатеричной системе. Завершает код символ «точка с запятой» (`;`). Десятичный код начинается символом «решетка» (`#`), а шестнадцатеричный – двумя – «решетка» и «латинская x» (`#x`).

Так, например, символ «<» определяется при помощи кода «`<`» («`<`» или «`<`»), а символ «>» – кода «`>`» («`>`» или «`>`»). Символ «знак зарегистрированной торговой марки» (®) имеет код «`®`» («`®`» или «`®`»), «знак copyright» (©) – код «`©`» («`©`» или «`©`»), а «неразрывный пробел» – «` `» («` `» или «` `»). Полный же список таких кодов имеется в любом справочнике

по языку HTML, например, в «HTML-справочник» Владимира Городулина – http://html.manual.ru/ или в «HTML в примерах» Александра Климова – http://webmaster.alexanderklimov.ru.

Все, с теорией закончили. Настала пора практики. В конце концов, учиться лучше всего на примерах. И дальше – по инструкции:

Ход работы

Шаг 1. Управление расположением текста на экране

1. В каталоге для выполнения лабораторных работ по HTML – `html` – создайте каталог `lr2` и сделайте его текущим.

В нем будут храниться документы HTML, создаваемые по ходу выполнения текущей лабораторной работы.

2. Из каталога Лабораторной работы № 1 – `lr1` – скопируйте файл `hello.html`. Если его там нет – создайте заново.

В результате в каталоге `lr2` должен быть файл `hello.html` – результат выполнения предыдущей Лабораторной работы.

3. Отредактируйте файл `hello.html`, убрав в нем информацию о цвете и расположив слова «`Всем`», «`огромный`» и «`привет!`» на разных строчках:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body>
Всем
огромный
привет!
</body>
</html>
```

4. Просмотрите его содержимое в браузере.

В результате текст в основном окне будет расположен в одну строку. Так происходит потому, что при отображении HTML-документов браузеры автоматически размещают текст на экране, не принимая во внимание встречающиеся в файле переводы строк и идущие подряд пробелы.

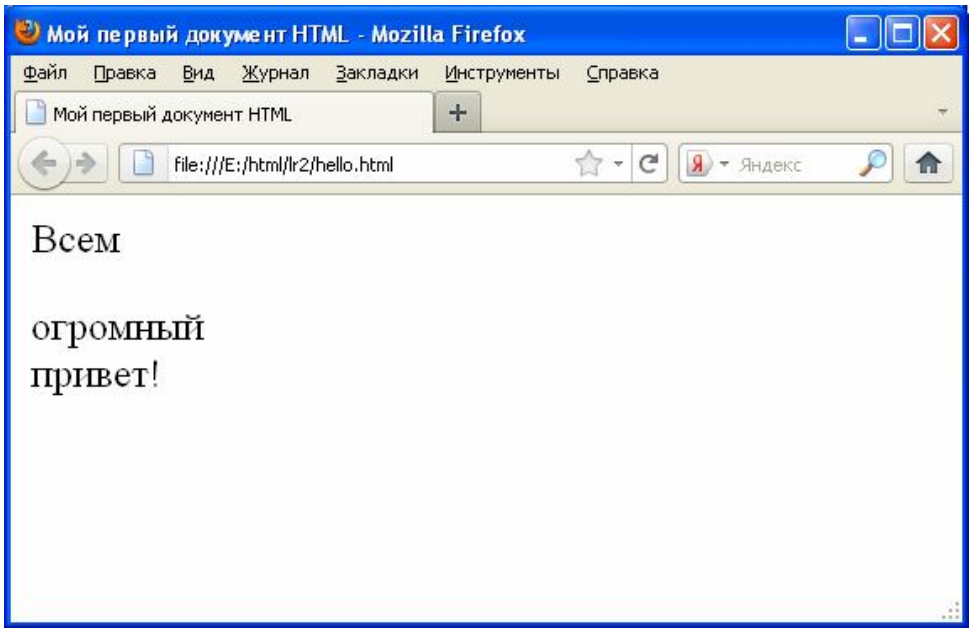
Шаг 2. Тэги перевода строки и абзаца

1. Внесите в текст файла HTML следующие изменения:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body>
Всем<p>
огромный<br>
привет!
</body>
</html>
```

2. Посмотрите обновленную WEB-страницу.

В результате она должна выглядеть примерно так:



Вид документа HTML изменился в результате того, что было добавлено два тега: `<br>` и `<p>`. Тэг перевода строки `<br>` отделяет строку от последующего текста или графики, а `<p>` – дополнительно добавляет пустую строку.

Шаг 3. Выделение фрагментов текста

1. Отредактируйте текст файла HTML следующим образом:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body>
<i><b>Всем</b></i> <i>огромный</i> <u>привет!</u>
</body>
</html>
```

2. Посмотрите обновленную Web-страницу.

Шаг 4. Использование стилей заголовка

1. Внесите в файл `hello.html` такие изменения:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body>
<h1>Всем</h1> <i>огромный</i> <u>привет!</u>
</body>
</html>
```

2. Посмотрите полученную Web-страницу.

Шаг 5. Задание размера текущего шрифта

1. Отредактируйте файл `hello.html` следующим образом:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body>
<font size=7>Всем</font>
огромный привет!
</body>
</html>
```

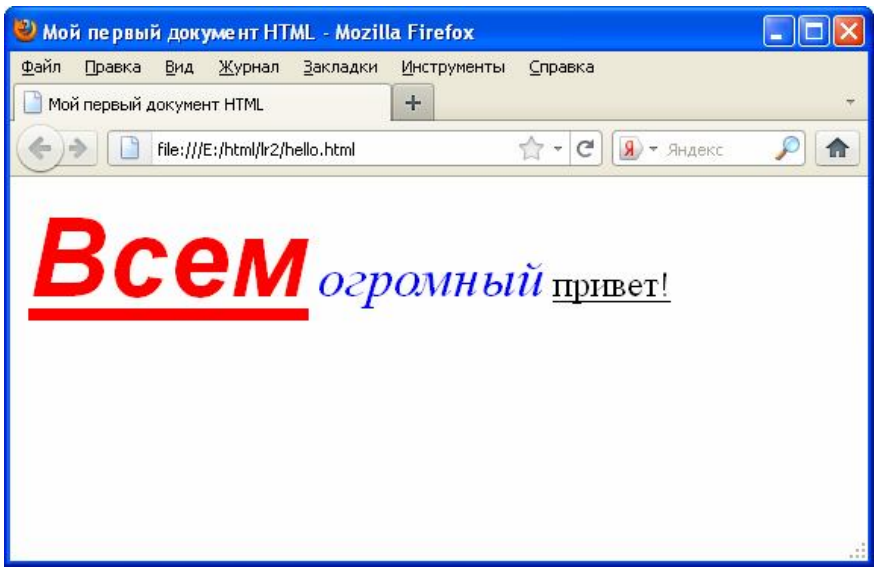
2. Самостоятельно измените размер шрифта для текста «`огромный привет!`», используя тэг `<font>`, а так же, тэги выделения фрагментов текста и тэги перевода строки и абзаца что бы получить такую Web-страницу:

Шаг 6. Гарнитура и цвет шрифта

1. Внесите в файл `hello.html` такие изменения:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body>
<u><i><b><font color="#ff0000" face="arial" size="7">
Всем</font></b></i></u> огромный привет!
</body>
</html>
```

2. Самостоятельно измените размер, цвет, гарнитуру, стиль текста документа, так что бы он выглядел следующим образом:



Шаг 7. Выравнивание текста по горизонтали

1. Отредактируйте файл `hello.html` следующим образом:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body>
<p align="center">
<font color="#008080" size="7"><b>Всем</b></font><br>
<font size="6"><i>огромный привет!</i></font>
</p>
</body>
</html>
```

2. Посмотрите полученный результат.
3. Проведите экспериментальную работу с созданным документом.

Меняя размеры окна, посмотрите, как браузер показывает текст с принудительным разрывом строк. Что происходит, когда окно уменьшается настолько, что в нем не может поместиться целиком даже одно слово?

Шаг 8. Отображение специальных символов

1. Отредактируйте файл `hello.html` так, чтобы слова «`огромный привет!`» всегда отображались в одной строке.

Для этого замените пробел между словами «`огромный`» и «`привет!`» на специальный символ «неразрывный пробел» – «` `». В результате файл `hello.html` должен иметь такое содержание:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body>
<p align="center">
<font color="#008080" size="7"><b>Всем</b></font><br>
<font size="6"><i>огромный привет!</i></font>
</p>
</body>
</html>
```

2. Посмотрите обновленную Web-страницу.

3. В конце Web-страницы `hello.html` перечислите все теги, которые были использованы при ее форматировании.

Для этого файл `hello.html` отредактируйте, например, следующим образом:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body>
<p align="center">
<font color="#008080" size="7"><b>Всем</b></font><br>
<font size="6"><i>огромный привет!</i></font>
</p>
Для форматирования этого текста были использованы
следующие теги:<br>
&lt;p&gt;,<br>
&#60;br&#62;,<br>
&#x3C;font&#x3E;,<br>
&lt;b&gt; и<br>
&lt;i&gt;.<br>
</body>
</html>
```

4. Посмотрите полученную Web-страницу.

Индивидуальные задания

1. В созданном ранее каталоге `lr2` создайте каталог `iz`.

В нем будут храниться документы HTML, создаваемые по ходу выполнения индивидуального задания текущей лабораторной работы.

2. В каталоге `iz` создайте свой документ HTML и сохраните его в файле под именем `index.html`.

Пусть это будет небольшое приветствие с рассказом, например, о себе, родителях, увлечениях и т. д. В заголовок окна поместите Вашу Фамилию Имя и Отчество.

3. В тетрадь для выполнения лабораторных работ занесите эскиз созданной Web-страницы и, соответствующей ей, текст HTML-документа.

Контрольные вопросы

1. Как браузеры обрабатывают символы пробелов и новых строк?
2. Как выполнить переход на новую строку на странице HTML?
3. Объясните назначение тега <p>.
4. Объясните назначение тега <br>.
5. Как установить полужирное начертание символов?
6. Как установить подчеркивание символов?
7. Как установить начертание символов курсив?
8. Как можно одновременно изменить несколько параметров начертания символов?
9. Как записываются вложенные теги?
10. Объясните структуру элемента документа HTML.
11. Объясните структуру тега.
12. Как отделяются друг от друга составляющие тега?
13. Для чего служат атрибуты тега?
14. Каков порядок следования атрибутов тега?
15. Сколько есть способов изменения размеров символов?
16. Для чего служат теги <h1> – <h6>?
17. Какой заголовок больше <h1> или <h6>?
18. Для чего служит тег <font>?
19. Какие атрибуты имеет тег <font>?
20. Как установить размер шрифта?
21. Как установить гарнитуру шрифта?
22. Как установить цвет выделенного фрагмента текста?
23. Как можно управлять горизонтальным выравниванием текста?
24. Какие значения может принимать атрибут align=?
25. С какими тегами может использоваться атрибут align=?
26. Что происходит с текстом при изменении размеров окна?
27. В чем разница между тегом
 и тегом <p>?
28. Как отобразить на Web-странице символ, которого нет на клавиатуре или который имеет специальное значение (например, «<»)?
29. Объясните структуру кода для отображения специальных символов.
30. Перечислите способы построения кодов для отображения специальных символов.
31. Сформулируйте и задайте свой вопрос по пройденному материалу.
32. Сформулируйте и внесите свое предложение по совершенствованию темы.

Лабораторная работа № 6
Гиперссылки и графика

Цель работы

Освоение и приобретение практических навыков создания и работы с гиперссылками, а также вставки графических изображений в HTML-документ и использования их в качестве гиперссылок.

Краткие теоретические сведения

При хранении вся информация может быть представлена либо в виде одного цельного блока (как один большой текстовый документ) или в виде нескольких независимых блоков, связанных между собой только специально оформленными ссылками. Хранение и представление информации в виде независимых частей (страниц), связанных между собой ссылками, называется *гипертекстом* или *гипертекстовым документом*. Ссылки между частями называются *гипертекстовыми ссылками* или просто *ссылками*. Ссылки могут указывать, как на отдельные части документа, так и на различные документы. Ссылки в пределах одного документа называются *внутренними*, а на другие документы – *внешними*. Внешние ссылки могут быть *относительными* – если указывают на документ в пределах одного сайта (или компьютера) и *абсолютными* – когда указывается протокол обмена и полный адрес документа в сети. Если будет указана только каталог, без имени конкретного файла, то предполагается, что это ссылка на сайт, и в этом каталоге будет открыт файл `index.html`, `index.htm`, `default.html`, `default.htm` или `index.php` (конкретное имя указывается в настройках Web-сервера).

Создать текстовую гиперссылку очень просто. Достаточно найти в HTML-документе фрагмент текста, который нужно превратить в гиперссылку, и заключить его в парный тег `<a>`. Интернет-адрес целевой Web-страницы указывают в атрибуте `href=` этого тега.

При помощи атрибута `name=` тега `<a>` указывается место в документе, на которое затем можно будет перейти. Другой способ указания точки перехода в документе – при помощи общего атрибута `id=`. Имя точки перехода, указанной при помощи атрибута `name=` или `id=`, в атрибуте `href=` должно обязательно начинаться символом «#». Например, если `name="label1"`, то `href="#label1"`. Атрибут `href=` тега `<a>` позволяет также создавать гиперссылки на конкретные метки во внешних файлах. В этом случае имя метки отделяется от имени файла символом «#». Например, `href="index.html#label1"` – ссылка на метку `label1` во внешнем файле `index.html`.

Правила отображения гиперссылок Web-обозревателем:

- при помещении курсора мыши на гиперссылку браузер меняет его форму на «указующий перст»;
- текст любых гиперссылок подчеркивается;
- обычные гиперссылки выделяются синим цветом;

• гиперссылки, по которым посетитель уже «ходил» (посещенные гиперссылки), выводятся темно-красным цветом;

• гиперссылка, по которой посетитель в данный момент щелкает (активная гиперссылка), выводится ярко-красным цветом.

Таково поведение по умолчанию.

Однако, цвет гиперссылок в документе HTML можно явно указать в теге `<body>` с помощью атрибутов:

• `link=` – просто ссылка,

• `alink=` – выделенная ссылка и

• `vlink=` – уже посещаемая ссылка.

С помощью необязательного атрибута `target=` тега `<a>` можно указать место загрузки ссылки:

• `target="_parent"` – в текущее окно браузера, или

• `target="_blank"` – в новое.

Атрибут `title=` тега `<a>` служит для создания подсказки, которая всплывает при выделении гиперссылки. Такая подсказка полезна, например, в случае, когда в качестве гиперссылки используется изображение.

Гиперссылки на адреса электронной почты указываются немного иначе, чем на другие документы, например, `<a href="mailto:pochta@mail.ru">Мой e-mail</a>`. Здесь «`mailto:`» записывается обязательно, и точно так, как указано. Подобным образом создаются гиперссылки на другие службы Internet, например, news-конференции, только вместо «`mailto:`» пишется «`news:`», а вместо адреса электронной почты – имя группы новостей.

В WWW сейчас используются всего три формата графических файлов:

• **GIF** (Graphics Interchange Format, формат обмена графикой) предназначен для хранения векторных (штриховых) изображений. Среди достоинств этого формата следует отметить то, что он позволяет задать для изображения «прозрачный» цвет, в одном GIF-файле можно хранить сразу несколько изображений, фактически – настоящий фильм (анимированный GIF). Недостаток у формата GIF всего один – он совершенно не годится для хранения полутоновых изображений (фотографий, картин и т. п.). Это обусловлено потерями качества при сжатии и тем, что GIF-изображения могут включать всего 256 цветов. Расширение имени файла формата GIF – `.GIF`.

• **JPEG** (Joint Photographic Experts Group, объединенная группа экспертов по фотографии) был разработан специально для хранения растровых (полутоновых) изображений. JPEG, в отличие от GIF, не ограничивает количество цветов у изображения, а реализованное в нем сжатие лучше всего подходит для полутоновых изображений. Однако он плохо справляется с штриховой графикой, не поддерживает «прозрачный» цвет и анимацию. Расширение имени файла формата JPEG – `.JPEG`, `JPG` или `JPE`.

• **PNG** (Portable Network Graphics, перемещаемая сетевая графика) был разрабатывался как замена формату GIF, а также, в некоторой степени, JPEG. В настоящее время он последовательно отвоевывает «жизненное пространство» у

этих форматов. К достоинствам формата PNG можно отнести возможность хранения как штриховых, так и полутоновых изображений и поддержку полупрозрачности. Недостаток всего один и некритичный – невозможность хранения анимации. Расширение имени файла формата PNG – `.PNG`.

Графические объекты в HTML-документ вставляются при помощи тега `<img>`. Атрибут `src=` является обязательным и указывает имя файла, содержащего графическое изображение. Выравнивание изображения выполняется при помощи атрибута `align=`. Так значение атрибута `align=left` выравнивает изображение по левому краю окна, `align=right` – по правому. Рамка вокруг изображения устанавливается при помощи атрибута `border=`. Ее толщина указывается в пикселях. Атрибуты `height=` и `width=` указывают размер изображения в пикселях по высоте и ширине, соответственно. Всегда желательно явно устанавливать эти атрибуты для ускорения загрузки страницы браузером. Атрибуты `hspace=` и `vspace=` указывают размер свободного места в пикселях слева и справа, и сверху и снизу от изображения, соответственно.

Язык HTML позволяет использовать в качестве содержимого гиперссылки любой фрагмент любого элемента, в том числе и графическое изображение, т. е. создавать изображение-гиперссылку. Для создания изображения-гиперссылки достаточно поместить внутрь тега `<a>` тег `<img>`.

Все, с теорией закончили. Настала пора практики. В конце концов, учиться лучше всего на примерах. И дальше – по инструкции:

Ход работы

Шаг 1. Создание гиперссылок в пределах одного каталога

1. В каталоге для выполнения лабораторных работ по HTML – `html`, создайте каталог `lr3` и сделайте его текущим.

2. Из каталога Лабораторной работы № 2 – `lr2`, скопируйте файл `hello.html`. Если его там нет – создайте заново.

В результате в каталоге `lr3` должен быть файл `hello.html` – результат выполнения предыдущей Лабораторной работы.

3. Создайте Web-страничку такого содержания и назовите ее `index.html`.

```
<html>
<head>
<title>Главная страница</title>
</head>
<body>
Приветствую Вас на своей страничке!<br><br>
Вот мое <a href="hello.html">приветствие</a>
</body>
</html>
```

4. Щелкните по ссылке на слове «`приветствие`».

В результате должен открыться документ `hello.html` – копия из предыдущей Лабораторной работы.

Шаг 2. Создание гиперссылок в пределах разных каталогов

1. В текущем каталоге создайте каталог `hello` и переместите туда файл `hello.html`.

2. В файле `index.html` замените строку:

«`Вот мое <a href ="hello.html"> приветствие<a>`»

на строку:

«`Вот мое <a href ="hello/hello.html"> приветствие<a>`».

Запись «`hello/hello.html`» говорит о том, что файл `hello.html` расположен во вложенном каталоге («этажом ниже») по отношению к текущему файлу – `index.html`.

3. Щелкните по ссылке на слове «`приветствие`».

В результате должен снова открыться файл `hello.html`.

4. В конце файла `hello.html` добавьте такую строку:

«`<a href="../index.html">Вернуться к главной странице</a>`».

Запись «`../index.html`» говорит о том, что файл `index.html` расположен в родительском каталоге («этажом выше») по отношению к текущему файлу – `hello.html`.

5. Проверьте работоспособность ссылок.

Шаг 3. Вставка рисунков

1. В текущем каталоге `lr3` создайте рисунок 50×50 точек (пикселей) – кружок **красного** цвета – и сохраните его в формате `.gif` в файле под именем `img.gif`.

Для этого, например:

• В операционной системе Linux:

а) запустите на выполнение графический редактор GIMP, выполнив команду **Меню ⇨ Графика ⇨ Редактор растровой графики GIMP**;

б) в окне **Редактор растровой графики GIMP** выполните команду **Файл ⇨ Создать**;

⚠ *Если некоторые элементы управления невидны – перетащите окно за заголовок по экрану монитора так, чтобы они стали видны.*

в) в раскрывшемся окне **Создать новое изображение** укажите **Размер изображения 50 × 50 точек растра** и щелкнете по кнопке **OK**;

г) в окне **… 50 × 50 – GIMP** выполните команду **Файл ⇨ Сохранить**;

д) в раскрывшемся окне **Сохранить изображение** укажите:

– в поле **Имя** – `img.gif`;

– при помощи раскрывающегося списка **Просмотреть другие папки** – `lr3`;

– при помощи раскрывающегося списка **Выберите тип файла** – **GIF image** и

– щелкните по кнопке **Сохранить**;

е) в раскрывшемся далее окне **Экспортировать файл** выберите значение переключателя по умолчанию – **Преобразовать в индексированное…** – и щелкните по кнопке **Экспорт**;

ж) в раскрывшемся далее окне **Сохранить как GIF** щелкните по кнопке **Сохранить**;

з) в окне **Панель инструментов** выберите инструмент **Эллиптическое выделение**;

и) при нажатой клавише **Shift** в поле рисования выделите окружность, перетащив указатель мыши по диагонали из одного угла в другой;

При перемещении по рисунку любого инструмента его точные координаты отображаются в низу окна – в строке состояния.

к) в окне **Панель инструментов** щелкните по инструменту **Цвет переднего плана**;

л) в раскрывшемся далее окне **Цвет переднего плана** в поле редактирования **HTML-разметка** введите код **красного** цвета – «`ff0000`» – и щелкните по кнопке **OK**;

в результате цвет переднего плана изменится на **красный**.

м) в окне **Панель инструментов** щелкните по инструменту **Плоская заливка:** ;

н) щелкните левой кнопкой мыши по выделенной области в поле рисования;

в результате выделенная область – круг – будет закрашена в **красный** цвет.

о) выполните команду **Файл ⇨ Сохранить** и подтвердите необходимость экспортирования (преобразования) изображения;

п) выполните команду **Файл ⇨ Выход**;

- В операционной системе Windows:

а) запустите на выполнение графический редактор Paint, например, выполнив команду **Пуск ⇨ Все программы ⇨ Стандартные ⇨ Paint**;

б) в раскрывшемся окне **Безымянный - Paint** выполните команду **Рисунок ⇨ Атрибуты…**;

в) в раскрывшемся далее окне диалога **Атрибуты** укажите в полях ввода **Ширина** и **Высота** – `50` **точек** и щелкните по кнопке **OK**;

г) в окне **Безымянный - Paint** выполните команду **Файл ⇨ Сохранить**;

д) в раскрывшемся окне диалога **Сохранить как** укажите:

– в поле редактирования **Папка** – `lr3`;
– в поле редактирования **Имя файла** – `img.gif`;
– в раскрывающемся списке **Тип файла** – **GIF (*.GIF)**;
– щелкните по кнопке **Сохранить**;

е) в раскрывшемся окне диалога подтвердите выполнение сохранения, щелкнув по кнопке **ДА**;

ж) в окне **img.GIF - Paint**:

1) выполните команду **Вид ⇨ Масштаб ⇨ Крупный**;

2) щелкните левой кнопкой мыши по **красному** цвету на палитре цветов внизу окна;

3) щелкните левой кнопкой мыши по инструменту рисования **Эллипс**;

4) укажите сплошную заливку без контура, щелкнув по последнему прямоугольнику ниже инструмента рисования **Эллипс**;

5) при нажатой клавише **Shift** нарисуйте в поле рисования кружок **красного** цвета, перетащив указатель мыши по диагонали из одного угла в другой;

При перемещении по рисунку любого инструмента его точные координаты отображаются в низу окна – в строке состояния.

з) выполните команду **Файл ⇨ Сохранить**;

и) выполните команду **Файл ⇨ Выход**;

2. Отредактируйте файл `index.html`, поместив после строки:

```
«Приветствую Вас на своей страничке!<br><br>»
```

тег вставки созданного рисунка `img.gif`:

```
«<img src="img.gif"><br><br>».
```

3. В браузере просмотрите файл `index.html`.

Если в окне браузера вместо картинки отображается стилизованный квадрат, то это значит, что:

- рисунок создан не в том формате или
- неправильно указано его расположение.

Для исправления ошибки необходимо еще раз внимательно выполнить все пункты текущего **Шага**.

Шаг 4. Использование рисунков в качестве гиперссылок

1. Строку файла `index.html`:

```
«<img src="img.gif"><br><br>»
```

отредактируйте следующим образом:

```
«<a href="hello/hello.html" title=" Перейти к
приветствию"><img src="img.gif"></a><br><br>».
```

2. Обновите файл `index.html` в окне браузера.

Обратите внимание, что в некоторых браузерах вокруг рисунка появляется рамка. Так происходит потому, что рисунок используется в качестве гиперссылки. Для того чтобы убрать рамку необходимо явно указать ее нулевую ширину с помощью атрибута `border="0"` тега `<img>`.

Шаг 5. Создание гиперссылок на объекты внутри документа

1. В файл `index.html` внесите такие изменения:

а) строку «`Приветствую Вас на своей страничке!`» пометьте при помощи атрибута `name="top"` тега `<a>`,

б) при помощи тега `<br>` в конце тела страницы, перед закрывающимся тегом `</body>`, вставьте 5 пустых строк,

в) перед закрывающимся тегом `</body>` вставьте ссылку на начало документа,

г) саму ссылку на начало страницы пометьте при помощи атрибута `id="bottom"`.

После внесенных изменений файл должен выглядеть следующим образом:

```
<html>
<head>
<title>Главная страница</title>
</head>
<body>
<a name="top">Приветствую Вас на своей
страничке!</a><br><br>
<a href="hello/hello.html" title="Перейти к
приветствию"><img src="img.gif"></a><br><br>
Вот мое <a href="hello/hello.html">приветствие</a>
<br>
<br>
<br>
<br>
<br>
<a id="bottom" href="#top">Перейти в начало документа</a>
</body>
</html>
```

2. Обновите файл `index.html` в окне браузера.

3. Уменьшите размер окна браузера по высоте на столько, чтобы файл `index.html` в него полностью не помещался.

4. При помощи вертикальной полосы прокрутки покажите окончание документа.

5. Проверьте работу ссылки на начало документа.

6. Отредактируйте файл `hello.html`, внеся в него, перед закрывающимся тегом `</p>`, гиперссылку на метку «`bottom`» в файле `index.html`.

В результате файл `hello.html` будет иметь такое содержание:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body>
<p align="center">
<font color="#008080" size="7"><b>Всем</b></font><br>
<font size="6"><i>огромный привет!</i></font><br>
<a href="../index.html">Вернуться к главной
странице</a><br>
<a href="../index.html#bottom">В конец главной
страницы</a>
</p>
</body>
</html>
```

7. Проверьте работу всех ссылок.

Шаг 6. Создание внешних гиперссылок

1. В файле `index.html` создайте гиперссылку на сайт Академии.

В результате он должен иметь такое содержание:

```
<html>
<head>
<title>Главная страница</title>
</head>
<body>
<a name="top">Приветствую Вас на своей
страничке!</a><br><br>
<a href="hello/hello.html" title="Перейти к
приветствию"><img src="img.gif"></a>
<br><br>
Вот мое <a href="hello/hello.html">приветствие</a>
<br>
<br>
<a href="http://www.kname.edu.ua/">Здесь я учусь</a>
<br>
<br>
<br>
<a id="bottom" href="#top">Перейти в начало документа</a>
</body>
</html>
```

2. Проверьте работу ссылки.

Если Ваш компьютер подключен к Интернету, будет открыты стартовая страница сайта Академии, если нет – выдана ошибка о том, что попытка соединения не удалась и запрашиваемую станицу отобразить невозможно.

Шаг 7. Создание гиперссылок на адреса электронной почты

1. В файле `index.html` создайте гиперссылку на произвольный адрес электронной почты или на свой – если он у Вас есть.

В результате он должен иметь такое содержание:

```
<html>
<head>
<title>Главная страница</title>
</head>
<body>
<a name="top">Приветствую Вас на своей
страничке!</a><br><br>
<a href="hello/hello.html" title="Перейти к
приветствию"><img src="img.gif"></a>
<br><br>
Вот мое <a href="hello/hello.html">приветствие</a>
<br>
<br>
<a href="http://www.kname.edu.ua/">Здесь я учусь</a>
<br>
<br>
```

```
<a href="mailto:borisu-p@ukr.net">Мой e-mail</a>
<br>
<br>
<a id="bottom" href="#top">Перейти в начало документа</a>
</body>
</html>
```

2. Проверьте работу ссылки.

Должна запуститься установленная по умолчанию в Вашей системе программа-клиент работы с электронной почтой.

Индивидуальные задания

1. В созданном ранее каталоге `lr3` создайте каталог `iz` и сделайте его текущим.

2. Создайте файл `index.html`, так чтобы:

- в заголовке окна была фраза «`Это я`»;
- Вашу Фамилию Имя и Отчество отформатировать заголовком 3-го уровня;
- содержал графическое изображение или Ваше фото;
- информация, например, о:

– дате рождения,
– месте проживания,
– родителях,
– любимых предметах,
– увлечениях и т. д.

была вынесена в отдельные файлы, а связь с основным выполнялась при помощи гиперссылок;

- добавьте так же гиперссылки на сайт Академии и адрес Вашей электронной почты.

3. В тетрадь для выполнения лабораторных работ занесите эскизы созданных Web-страниц и, соответствующие им, тексты HTML-документов.

Контрольные вопросы

1. Что такое гипертекстовый документ?
2. Что такое гиперссылка?
3. Что такое внутренняя ссылка?
4. Что такое внешняя ссылка?
5. Как создать гиперссылку на определенное место внутри другого HTML-документа?
6. Какие есть способы создания меток в HTML-документе для создания на них гиперссылок?
7. Что такое относительная ссылка и как она создается?
8. Как указать гиперссылку на файл в текущем каталоге?
9. Как указать гиперссылку на файл в родительском каталоге (уровнем выше)?
10. Как указать гиперссылку на файл во вложенном каталоге (уровнем ниже)?

11. Какой документ откроет браузер, если в гиперссылке указан только сайт (каталог), без указания конкретного документа?
12. Как указать цвет гиперссылки?
13. Как указать цвет выделенной гиперссылки?
14. Как указать цвет уже посещаемой гиперссылки?
15. Как указать загрузку гиперссылки в текущее окно браузера?
16. Как указать загрузку гиперссылки в новое окно браузера?
17. Как указать подсказку, когда в качестве гиперссылки используется изображение?
18. Как создать гиперссылку на адрес электронной почты?
19. Как создать гиперссылку на news-конференцию?
20. Какие форматы файлов изображений можно вставлять в HTML-документ?
21. Как создать графическое изображение в операционной системе Linux?
22. Как создать графическое изображение в операционной системе Windows?
23. Как вставить графическое изображение в HTML-документ?
24. Как указывается имя файла, содержащего графическое изображение?
25. Как выровнять изображение полевому краю окна браузера?
26. Как выровнять изображение правому краю окна браузера?
27. Как установить, или наоборот – убрать, рамку вокруг изображения?
28. Как установить размер свободного места вокруг изображения?
29. Как указать размер изображения?
30. Как использовать изображение в качестве гиперссылки?
31. Сформулируйте и задайте свой вопрос по пройденному материалу.
32. Сформулируйте и внесите свое предложение по совершенствованию темы.

Лабораторная работа № 7
Таблицы

Цель работы

Освоение и приобретение практических навыков работы с таблицами в HTML-документах.

Краткие теоретические сведения

Таблицы в HTML-документе создаются за четыре этапа:

- **На первом этапе** в HTML-коде с помощью парного тега `<table>` формируют саму таблицу:

```
<table>
</table>.
```

- **На втором этапе** формируют строки таблицы. Для этого предусмотрены парные теги `<tr>` (Table Row); каждый такой тег создает отдельную строку. Теги `<tr>` помещают внутрь тега `<table>`.

- **На третьем этапе** создают ячейки таблицы, для чего используют парные теги `<td>` (Table Data) и `<th>` (Table Header). Тег `<td>` создает обычную ячейку, а тег `<th>` – ячейку заголовка, в которой будет помещаться «шапка» соответствующего столбца таблицы. Теги `<td>` и `<th>` помещают в теги `<tr>`, создающие строки таблицы, в которых должны находиться эти ячейки.

- **На четвертом**, последнем, этапе указывают содержимое ячеек, которое помещают в соответствующие теги `<td>` и `<th>`.

Ячейки таблицы HTML должны иметь хоть какие-то содержимое, иначе Web-обозреватель может их вообще не отобразить. Если же какая-то ячейка должна быть пустой, в нее следует поместить неотображаемый символ «неразрывный пробел» (` `). Если одна или несколько ячеек, располагающихся в конце какой-либо строки, не содержат данных, то их описание может быть опущено, а браузер автоматически добавит требуемое количество пустых ячеек. Отсюда следует, что построение таблиц, в которых в разных строчках располагается различное количество столбцов одного и того же размера, не разрешается.

По умолчанию размер таблицы устанавливается в зависимости от ее содержания так, чтобы она отображалась наилучшим образом. С помощью атрибутов `width=` и `height=` можно явно указать размеры ячейки, строки и всей таблицы. Их значения могут быть указаны либо в пикселях, либо в процентах. Горизонтальное выравнивание в таблицах устанавливается атрибутом `align=`, а вертикальное – `valign=`.

Различие в использовании тегов `<td>` и `<th>` заключается лишь в типе шрифта, используемого по умолчанию для отображения содержимого ячеек, а также расположению данных внутри ячейки. Содержимое ячеек типа `<th>` отображается полужирным (**Bold**) шрифтом и располагается по центру (`align="center"`, `valign="middle"`) Ячейки, определенные тегом `<td>`

по умолчанию отображают данные, выровненные влево (align="left") и посередине (valign="middle") в вертикальном направлении.

Завершающие теги </td>, </th> и </tr> могут быть опущены. В этом случае концом описания строки или ячейки является начало следующей строки или ячейки, или конец таблицы. Завершающий тег таблицы </table> не может быть опущен.

Таблица может иметь заголовок, который заключается в пару тегов <caption>...</caption>. Описание заголовка таблицы должно располагаться внутри тегов <table>...</table> до первого <tr>. По умолчанию текст заголовка таблицы располагается над ней – align="top" – и центрируется по горизонтали. Можно так же указать расположение заголовка под таблицей – align="bottom".

В таблицах HTML-документов смежные ячейки можно объединять в одну, как по горизонтали, так и по вертикали. Специально для этого теги <td> и <th> поддерживают два необязательных атрибута. Первый – colspan= – объединяет ячейки по горизонтали, второй – rowspan= – по вертикали.

Чтобы объединить несколько ячеек по горизонтали в одну, нужно выполнить следующие шаги:

1. Найти в коде HTML тег <td> (<th>), соответствующий первой из объединяемых ячеек (если считать ячейки слева направо).

2. Вписать в него атрибут colspan= и присвоить ему количество объединяемых ячеек, считая и самую первую из них.

3. Удалить теги <td> (<th>), создающие остальные объединяемые ячейки данной строки.

Объединить ячейки по вертикали чуть сложнее. Вот шаги, которые нужно для этого выполнить:

1. Найти в коде HTML строку (тег <tr>), в которой находится первая из объединяемых ячеек (если считать строки сверху вниз).

2. Найти в коде этой строки тег <td> (<th>), соответствующий первой из объединяемых ячеек.

3. Вписать в него атрибут rowspan= и присвоить ему количество объединяемых ячеек, считая и самую первую из них.

4. Просмотреть последующие строки и удалить из них теги <td> (<th>), создающие остальные объединяемые ячейки.

HTML-код, создающий таблицы, может показаться несколько громоздким. Но это плата за исключительную гибкость таблиц HTML. В них можно поместить любое содержимое, например: абзацы, заголовки, изображения, аудио- и видеоролики и даже другие таблицы.

Таблицы в HTML используются не только традиционно, как метод представления данных, но и как средство разметки Web-страниц. Так плавающие, или встроенные, фреймы, которые создаются при помощи тега <iframe>, позволяет вставлять один HTML-документ в другой наподобие матрешек. В частности можно «загружать» («подгружать») в ячейки таблицы

необходимые HTML-документы. Если браузер пользователя не поддерживает плавающих фреймов, то соответствующее сообщение необходимо поместить внутри контейнера `<iframe>...</iframe>`. Размеры плавающего фрейма указываются в атрибутах `width=` и `height=`, а в `frameborder=` – отображать (`1`) или не отображать (`0`) рамку вокруг него.

Все, с теорией закончили. Настала пора практики. В конце концов, учиться лучше всего на примерах. И дальше – по инструкции:

Ход работы

Шаг 1. Создание простейшей таблицы

1. В каталоге для выполнения лабораторных работ по HTML – `html`, создайте каталог `lr4` и сделайте его текущим.

2. На основе созданного в первой Лабораторной работе шаблона Web-страницы `template.html`, или создайте заново, файл `tables.html`, который содержит простейшую таблицу, состоящую из 2-х строк и 3-х столбиков, в каждой ячейке которой записаны ее «координаты», например, «`1x1`» для – 1-й ячейки 1-й строки, «`1x2`» – для 2-й ячейки 1-й строки, и т. д.:

```
<html>
<head>
<title>Таблицы</title>
</head>
<body>
<table>
<tr>
<td>1x1</td><td>1x2</td><td>1x3</td>
</tr>
<tr>
<td>2x1</td><td>2x2</td><td>2x3</td>
</tr>
</table>
</body>
</html>
```

3. Просмотрите созданную таблицу в браузере.

Если не было допущено ошибок, то она должна выглядеть примерно так:

Столь незатейливый вид таблицы обусловлен тем, что так выглядят все таблица с параметрами тегов, принятыми по умолчанию.

Шаг 2. Добавление рамок

1. В файле `tables.html` отредактируйте тег создания таблицы `<table>`, добавив к нему атрибут управления рамкой `border=`, указав ее толщину в `1` пиксель:

«`<table border="1">`».

2. Сохраните отредактированный файл `tables.html`, обновите содержимое окна браузера, и Вы увидите результат.

Шаг 3. Пустые ячейки

1. Отредактируйте файл `tables.html`, «пропустив» в нем 2-ю ячейку в 1-й строке, т. е. тег описания 1-й строки будет выглядеть следующим образом:

«`<td>1x1</td><td>1x3</td>`».

2. Посмотрите полученный результат.

Он должен выглядеть примерно так:

1x1	1x3	
2x1	2x2	2x3

3. Снова отредактируйте файл `tables.html`, вставив в нужном месте «пустую» ячейку 1x2.

В результате тег описания 1-й строки должен выглядеть следующим образом:

«`<td>1x1</td><td></td><td>1x3</td>`».

4. Посмотрите изменения в браузере.

В результате место под ячейку 1x2 будет выделено, но рамки вокруг нее не будет.

5. Для тог, чтобы вокруг пустой ячейки отображалась рамка, она должна содержать, по крайней мере, один из неотображаемых символов. Например, «неразрывный пробел» – «` `». В результате описание 1-й строки должно выглядеть следующим образом:

«`<td>1x1</td><td> </td><td>1x3</td>`».

Шаг 4. Объединение ячеек

1. Отредактируйте файл `tables.html` так, чтобы 1-я и 2-я ячейки 1-го ряда были объединены.

Для этого необходимо отредактировать описание 1-й строки следующим образом:

«`<td colspan="2">1x1</td><td>1x2</td>`».

2. Посмотрите полученный результат в браузере.

Он должен выглядеть примерно так:

1x1		1x2
2x1	2x2	2x3

3. Отредактируйте описание 1-й строки следующим образом – т. е. «забудьте» удалить ячейку 1x3:

«<td colspan="2">1x1</td><td>1x2</td><td>1x3</td>».

Посмотрите полученный результат.

Он должен выглядеть примерно так:

1x1	1x2	1x3
2x1	2x2	2x3

4. Самостоятельно, при помощи атрибута `rowspan=`, объедините ячейки 1x3 и 2x3так, чтобы получить такую таблицу:

1x1	1x2	1x3
2x1	2x2	

Шаг 5. Вложенные таблицы

1. Создайте таблицу, состоящую из 1-й строки и 2-х столбиков. Во второй столбик поместите таблицу, состоящую из 2-х строк и 1-го столбика. Для этого отредактируйте файл `tables.html` следующим образом:

```
<html>
<head>
<title>Таблицы</title>
</head>
<body>
<table border="1">
<tr>
<td>1+2</td>
<td>
<table border="1">
<tr><td>1x2</td></tr>
<tr><td>2x2</td></tr>
</table>
</td>
</tr>
</table>
</body>
</html>
```

2. Посмотрите полученный результат.

Он должен выглядеть примерно так:

1+2	1x2
	2x2

Вложенные таблицы и объединение ячеек – два способа решения одной задачи. Какой из них предпочесть, зависит о решаемой задачи. Объединение ячеек – более простой метод, и применяется в более простых ситуациях; вложенные таблицы позволяют более гибко конструировать сложные таблицы.

Шаг 6. Добавление заголовков

1. Отредактируйте файл `tables.html`, добавив к таблице заголовок «Баланс», а 1-му и 2-му столбикам – «Приход» и «Расход», соответственно.

В результате он должен иметь такое содержание:

```
<html>
<head>
<title>Таблицы</title>
</head>
<body>
<table border="1">
<caption><h3>Баланс</h3></caption>
<tr>
<th>Приход</th><th>Расход</th>
</tr>
<tr>
<td>1000</td><td>2000</td>
</tr>
</table>
</body>
</html>
```

2. Посмотрите полученный результат в браузере.

Он должен выглядеть примерно так:

Заголовок таблицы добавляется при помощи тега `<caption>...</caption>`. По умолчанию он отображается основным текстом таблицы. Для того, чтобы его выделить был использован тег заголовка `<h3>...</h3>`. Заголовки столбиков выводятся при помощи тега `<th>...</th>`. По умолчанию он выводится полужирным начертанием, выровненным по середине ячейки. Содержимое ячеек созданных при помощи тега `<td>...</td>` выводится нормальным шрифтом, выровненным по левому краю ячейки. Обратите также внимание, что размер ячеек был подобран под их наибольшее содержание.

Шаг 7. Использование таблиц для разметки Web-страниц

Создайте миниWeb-сайт для отображения личных страниц Лизы, Иры, Кати, Светы и Наташи. Он должен иметь такую структуру:
В левом поле располагается меню, при помощи которого в правое нижнее поле будут загружаться персональные страницы. В правом верхнем поле должен отображаться логотип.

<table>
<tr><td rowspan="2">Меню</td><td>Логотип</td></tr>
<tr><td>Содержание</td></tr>
</table>

1. Создайте Web-страницу `liza.html`, с информацией о Лизе, например, такого содержания:

```
<html>
  <head>
    <title>Лиза</title>
  </head>
  <body>
    Меня зовут Лиза
  </body>
</html>
```

2. Создайте аналогичные HTML-документы для следующих 4-х женских имен: Ира, Катя, Света и Наташа, соответственно, `ira.html`, `katya.html`, `sveta.html` и `natasha.html`.

3. Создайте файл `menu.html`, содержащий гиперссылки на страницы HTML с информацией о Лизе, Ире, Кате, Свете и Наташе:

```
<html>
  <head>
    <title>Меню</title>
  </head>
  <body>
    <h3>Меню</h3>
    <a href="liza.html" target="content">Лиза</a><br>
    <a href="ira.html" target="content">Ира</a><br>
    <a href="katya.html" target="content">Катя</a><br>
    <a href="sveta.html" target="content">Света</a><br>
    <a href="natasha.html" target="content">Наташа</a>
  </body>
</html>
```

4. При помощи графического редактора создайте логотип размером 80×80 точек (пикселей), например, кружок зеленого цвета, и сохраните его в формате `.gif` в файле под именем `logo.gif`.

5. Отредактируйте файл `tables.html` так, чтобы он имел следующее содержание:

```
<html>
  <head>
    <title>Таблицы</title>
  </head>
  <body>
    <table height="100%">
      <tr valign="top">
        <td>
          <iframe src="menu.html" width="80"
height="100%" frameborder="0">
            <font color="ff0000">Чтобы увидеть эту
страницу обновите браузер</font>
          </iframe>
        </td>
        <td>
          <table>
            <tr><td>
              <img src="logo.gif">
            </td></tr>
            <tr><td>
              <iframe src="liza.html" name="content"
frameborder="0"></iframe>
            </td></tr>
          </table>
        </td>
      </tr>
    </table>
  </body>
</html>
```

6. Проверьте работу созданного миниWeb-сайта.

7. Самостоятельно удалите, или измените, значение атрибутов `width=`, `height=` и `valign=`, которые служат для явной установки ширины, высоты и вертикального выравнивания, соответственно, и посмотрите, как «поплывет» таблица при изменении размеров окна браузера.

Индивидуальные задания

1. Создайте каталог `iz` и сделайте его текущим.
2. Создайте Web-страницу `result.html`, которая содержит таблицу с результатами предыдущей сессии. Каждая строка таблицы должна иметь такие колонки (как в зачетной книжке):
   - Номер по порядку,
   - Наименование учебной дисциплины,
   - Объем часов по учебному плану,
   - Фамилия Имя Отчество преподавателя,
   - Дата проведения,
   - Экзаменационная оценка или информацию о зачете.
3. Создайте личный миниWeb-сайт, на котором будете отображать информацию о:
   - себе,
   - родителях,

- друзьях,
- любимых предметах,
- увлечениях,
- результатах предыдущей сессии (файл `result.html`)
- и т. д.

Разметка главной страница должна быть выполнена с помощью таблицы. Она должна содержать минимум 3 области: меню, логотип и содержание.

4. В тетрадь для выполнения лабораторных работ занесите эскизы созданных Web-страниц и, соответствующие им, тексты HTML-документов.

Контрольные вопросы

1. Как создать таблицу в HTML-документе?
2. Как должна начинаться и заканчиваться таблица в HTML-документе?
3. Из чего состоит таблица?
4. Как начинаются и завершаются строки таблицы?
5. Как создаются ячейки в строке?
6. Для чего служит тег `<td>`?
7. Для чего служит тег `<th>`?
8. В чем отличие тега `<th>` от `<td>`?
9. Где могут появляться, а где нет, теги `<td>` и `<th>`?
10. Обязательно ли наличие тегов `</td>`, `</th>` и `</tr>`?
11. Обязательно ли наличие тега `</table>`?
12. Как определяется количество строк в таблице?
13. Как определяется количество столбцов в таблице?
14. Как создаются пустые ячейки внутри строки?
15. Как создаются пустые ячейки в конце строки?
16. Можно ли создать таблицу, в строках которой расположено различное количество столбцов одинаковой ширины?
17. Как создается заголовок таблицы?
18. Где может располагаться описание заголовка таблицы?
19. Где по умолчанию располагается заголовок таблицы?
20. Где может располагаться заголовок таблицы?
21. Как устанавливается размер таблицы?
22. Какие атрибуты явно задают размер таблицы?
23. В каких единицах устанавливаются размеры таблицы?
24. Как явно установит горизонтальное и вертикальное выравнивание в таблице?
25. Как объединять столбцы таблицы?
26. Как объединять строки таблицы?
27. Допускается ли вложенность таблиц в документах HTML?
28. Как используются таблицы в HTML-документах?
29. Для чего служит тег `<iframe>`, какие он имеет атрибуты?
30. Как предотвратить «плавание» таблицы при изменении размеров окна браузера?
31. Сформулируйте и задайте свой вопрос по пройденному материалу.
32. Сформулируйте и внесите свое предложение по совершенствованию темы.

Лабораторная работа № 8
Каскадные таблицы стилей

Цель работы

Освоение и приобретение практических навыков работы с каскадными таблицами стилей для управления представлением содержимого HTML-документов.

Краткие теоретические сведения

Каскадные таблицы стилей (англ. *CSS*, *Cascading Style Sheets*) – это технология описания внешнего вида документа, оформленного языком гипертекстовой разметки. CSS позволяют разделить смысловое содержание Web-страниц и их оформление. Если HTML предоставляет информацию о структуре документа, то CSS определяет, как он должен выглядеть.

Стиль – это поименованная совокупность параметров форматирования, применяемых к элементу Web-страницы, и определяющих способ его отображения. Он включает такие элементы дизайна как параметры отображения текста, фона, расположение объектов на странице и т д.

Так как в одном HTML-документе встречается множество различных элементов, каждый из которых может иметь свой стиль оформления, то набор таких стилей (*правил*) называют *таблицей стилей*.

Каскадирование – это порядок применения различных стилей к элементам Web-страницы. Оно применяется в том случае, если для одного элемента разными способами задано несколько различных стилей. Другой аспект каскадирования – *наследование* (англ. *inheritance*), – означает, что если не указано иное, то конкретный стиль будет применен ко всем дочерним элементами гипертекстового документа. Например, если применить определенный цвет текста в теге `<div>`, то все теги внутри этого блока будут отображаться этим же цветом.

Основной составляющей каскадных таблиц стилей является *правило*, которое описывает внешний вид одного или группы элементов HTML. Структура правила CSS представлена на рис. 2.

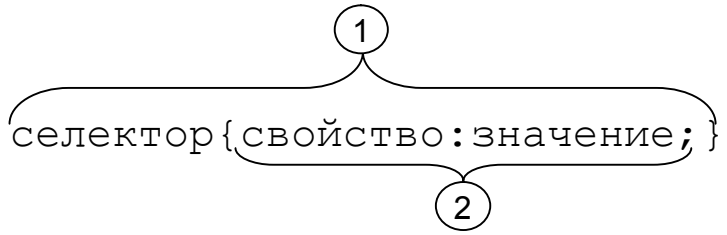


Рис. 2. Структура правила CSS

Правило (англ. *Rule*) (поз. 1 рис. 2) состоит из *селектора* (англ. *Selector*), который всегда располагается слева, и блока объявления (англ. *Declaration*), который заключается в фигурные скобки и следует непосредственно за селектором (поз. 2 рис. 2). Каждое объявление, в свою очередь, состоит из *свойства* (англ. *Property*) и его *значения* (англ. *Value*). Именно свойства и их

значения и определяет вид стиля, который будет применяться к элементу Web-страницы. Правило может содержать несколько объявлений, отделенных точкой с запятой. После последнего объявления точку с запятой можно не ставить. Описание правил CSS похоже на описание атрибутов тегов HTML, только разделяются они не пробелами, а точкой с запятой, и присваиваются значения не знаком равенства, а двоеточием. Правило можно набирать как прописными, так и строчными буквами. Порядок перечисления селекторов в таблице и свойств в объявлении не регламентирован.

Селектор в CSS служит для того, чтобы в HTML-документе однозначно определить тот элемент, к которому следует применить данное правило. В каскадных таблицах стилей предусмотрено несколько типов селекторов. Наиболее распространенные из них такие:

- *Селектор по элементу*, кода в качестве селектора используется имя тега.
- *Селектор по классу*. Используется когда правило необходимо применить не ко всем элементам (тегам), а только к определенным. В определении селектора по классу его имя обязательно предваряется символом точки.
- *Селектор по идентификатору*. Используется когда правило необходимо применить только к одному, вполне определенному элементу (тегу) на Web-странице. В определении селектора по идентификатору перед его именем ставят символ «решетка» (#).
- *Контекстный селектор* позволяет указать, что правило CSS необходимо применить для элемента находящегося исключительно внутри другого элемента. Имена элементов, указываемых в селекторе, отделяются друг от друга пробелам и размещаются в порядке их расположения в дереве элементов.
- *Псевдоклассы* позволяют привязать стиль к элементам Web-страницы на основе их состояния, например, наведен на них указатель мыши или нет. Их записывают сразу после основного селектора без всяких пробелов. Имя псевдокласса пишется через двоеточие (`:`) от имени элемента, внутри которого следует использовать этот псевдокласс. В CSS существует следующие псевдоклассы, позволяющих работать с гиперссылками: `:link` – непосещенная гиперссылка; `:visited` – посещенная гиперссылка; `:active` – гиперссылка, на которой посетитель в данный момент щелкает мышью; `:focus` – гиперссылка, имеющая фокус ввода; `:hover` – гиперссылка, на которую наведен указатель мыши.

Псевдоклассы гиперссылок применяются только совместно со стилями, задающими параметры для гиперссылок.

Группировка (*grouping*) – это объединение одинаковых деклараций или отдельных свойств с целью уменьшения общего объема кода. В CSS существует два типа группировок:

- объединение *селекторов* с одинаковыми объявлениями и
- объединение значений родственных *свойств*.

При группировке *селекторов* с одинаковыми объявлениями они разделяются запятой. При группировке значений родственных *свойств* таблица стилей становится более компактной, но предъявляются более жесткие требования к описанию правил. Группировка значений родственных свойств

может применяться для таких свойств, как padding:, font:, border:, background: и т. д.

Для указания значений свойств в CSS предусмотрены различные единицы измерения. Все они делятся на два больших класса: *относительные* и *абсолютные*. Абсолютные единицы измерения имеют фиксированный размер, не зависящий от других величин. К ним, например, относятся: mm – миллиметр, cm – сантиметр, in – дюйм, pt – пункт и pc – пика. Относительные устанавливают размер относительно какой-то другой единицы, например, точки на экране монитора, размер которой в свою очередь зависит от разрешения. Их всего четыре: % – процент от размера родительского элемента, px – пиксель, равен одной точке на экране монитора, em – размер (ширина) буквы «m» текущего шрифта и ex – размер (высота) буквы «x» текущего шрифта.

Существует три способа применения CSS к документу HTML:

- *Встраивание* (англ. *Inline Style*). Он предоставляет максимальный контроль над всеми элементами Web-страницы.
- *Внедрение* (англ. *Embedding Style*). Внедрение позволяет управлять стилями страницы целиком. Правила заключают в парный тег <style> и помещают в секцию заголовка HTML-документа (<head>...</head>).
- *Связывание* (англ. *Linking Style* или *External Style*). Связанная таблица стилей позволяет вынести описание стилей во внешний файл, ссылаясь на который можно контролировать отображение всех страниц сайта.

При возникновении конфликта применения к одному и тому же элементу Web-страницы нескольких стилей из различных источников он решается согласно правилам каскадирования, которые схематично можно представить так: *связанные стили –> внедренные стили –> встроенные стили*. Так же более конкретные стили имеют приоритет над менее конкретными. Это значит, что правило с селектором по идентификатору имеет приоритет над правилом с селектором по классу, которое, в сою очередь, имеет приоритет над правилом с селектором по элементу Web-страницы. Схематично это можно представить так: *селектор по элементу –> селектор по классу –> селектор по идентификатору*. Если же необходимо «нарушить» установленные правила каскадирования, то требуемое свойство надо сделать *важным*. Для этого достаточно после его значения поставить слово «!important».

Каскадные таблицы стилей помимо оформления содержания Web-страниц так же широко используются для их разметки. В зависимости от способа вывода все элементы Web-страницы делятся на следующие две категории:

- *строчные* (они же *встроенные*, англ. inline) – выводятся в строке друг за другом вдоль координаты x, пока не закончится строка, осуществляя затем переход на следующую (например, font, a, img, u, span и др.) и
- *блочные* (*блоки*, англ. block) – выводятся друг под другом вдоль координаты y (например, p, h1 ÷ h6, div и т. д.).

Свойство display: определяет, как элемент должен быть показан на Web-странице. Список возможных значений этого свойства, понимаемый разными Web-браузерами очень короткий: block – элемент показывается как блочный; inline – элемент отображается как встроенный; list-item – элемент

выводится как блочный и добавляется маркер списка; none – временно удаляет элемент из HTML-документа. Значение по умолчанию для свойства display: зависит от конкретного элемента Web-страницы. Так, для абзаца значением по умолчанию будет block, а для графического изображения – inline.

Свойство position: в сочетании со свойствами left:, top:, right:, bottom: определяет схему позиционирования, т. е. позволяет управлять положением блочных элементов относительно окна браузера или других объектов на Web-странице. Оно может принимать четыре значения, которые соответствуют основным схемам позиционирования: static – элементы отображаются как обычно (нормальный поток); relative – относительное позиционирование (относительно нормального потока); absolute – абсолютное позиционирование, такой блок не перемещается при прокрутке.

Плавающие блоки в CSS реализуются с помощью свойства float:, которое может принимать следующие значения: none – блок не перемещается, т. е. позиционируется согласно алгоритму нормального потока; left – блок перемещается влево, остальное содержимое документа будет обтекать его вдоль правой стороны, начиная с самого верха; right – блок перемещается вправо, остальное содержимое документа будет обтекать его вдоль левой стороны, начиная с самого верха.

В CSS предусмотрено специальное свойство clear:, которое позволяет однозначно указать, что данный блочный элемент в любом случае должен располагаться ниже плавающих контейнеров, предшествующих ему. Оно имеет четыре доступных значений: none – обычное поведение, т. е. обтекание элемента происходит как задано с помощью свойства float:; left – элемент Web-страницы должен располагаться ниже всех элементов, для которых свойство float: имеет значение left; right – элемент Web-страницы должен располагаться ниже всех элементов, для которых свойство float: имеет значение right; both – элемент Web-страницы должен располагаться ниже всех элементов, для которых свойство float: имеет значение left или right.

Каскадные таблицы стилей помимо основного текста – правил – могут содержать и *комментарии*. Они ни как не влияют на отображение документа, а служат лишь для пояснения отдельных его фрагментов. Комментарий в CSS начинается с последовательности символов «/*» а заканчивается – «*/», например, /* Это комментарий */.

Более подробно список всех свойств, а также их значений, можно посмотреть в любом справочнике по CSS, например, в Справочнике «Стили CSS» Влада Мержевича – htmlbook.ru. Исчерпывающая же, официальная информация о спецификации CSS всегда доступна по адресу http://www.w3.org/Style/CSS/.

Все, с теорией закончили. Настала пора практики. В конце концов, учиться лучше всего на примерах. И дальше – по инструкции:

Ход работы

Шаг 1. Встроенные стили (Inline Styles)

1. В каталоге для выполнения лабораторных работ по HTML – `html`, создайте каталог `lr5` и сделайте его текущим.

2. Создайте заготовку (шаблон) Web-страницы `liza_tmplt.html`, с персональной информацией о Лизе, например, такого содержания:

```
<html>
  <head>
    <title>Лиза</title>
  </head>
  <body>
    <p>Приветствую Вас на своей страничке!</p>
    <h1>Меня зовут Лиза</h1>
    <p>Мне 18 лет.</p>
    <p>Я учусь в <a href=http://www.kname.edu.ua/
                   title="Здесь я учусь">ХНАГХ</a>
      на факультете Экономики
      и предпринимательства.<br>
      Моя будущая специальность - Учет и Аудит.</p>
    <p>Моими любимыми дисциплинами являются
      Бухгалтерский учет и Информатика.</p>
    <p>В свободное время я занимаюсь танцами
      и рисованием.</p>
    <p>Ваши сообщения можете отправлять на мой
       e-mail: <a href="mailto:lize@ukr.net">
                lize@ukr.net</a></p>
  </body>
</html>
```

3. Просмотрите созданную Web-страницу в браузере.

Это будет простая Web-страничка, без какой-либо разметки и форматирования. В ней указаны лишь основные структурные единицы – абзацы, заголовки, гиперссылки и т. д., – т. е. только содержание (контент).

4. Откройте в текстовом редакторе файл `liza_tmplt.html` (если он еще не открыт) и сохраните его под именем `liza_0.html`.

5. Отредактируйте файл `liza_0.html`, установив при помощи встроенных стилей такие параметры форматирования:

а) текст на всей странице: Arial, если такой отсутствует – любой рубленный (sans-serif), цвет – тускло-серый (dimgray – `#696969`);

б) абзац «Приветствую Вас …»: символы – ВСЕ ПРОПИСНЫЕ, цвет – **зеленый**, начертание – **полужирное**;

в) заголовок_1 «Меня зовут ..»: с тенью размером 1 пиксель, цвет – `#CDD9DB`;

г) весь остальной текст на странице: размер – 14 пунктов.

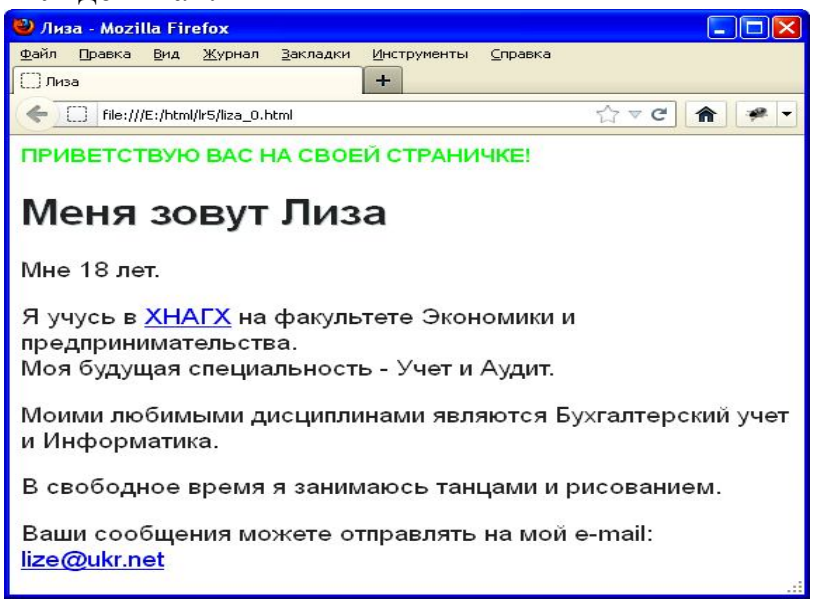
В результате он должен иметь следующее содержание:

```
<html>
  <head>
    <title>Лиза</title>
  </head>
  <body style="font-family:Arial, sans-serif;
    color:dimgray;">
    <p style="text-transform:uppercase;
      color:#00FF00; font-weight:bold;">
      Приветствую Вас на своей страничке!</p>
    <h1 style="text-shadow:1px 1px #CDD9DB;">
      Меня зовут Лиза</h1>
    <div style="font-size:14pt;">
      <p>Мне 18 лет.</p>
      <p>Я учусь в <a href=http://www.kname.edu.ua/
                    title="Здесь я учусь">ХНАГХ</a>
        на факультете Экономики
        и предпринимательства.<br>
        Моя будущая специальность - Учет и Аудит.</p>
      <p>Моими любимыми дисциплинами являются
        Бухгалтерский учет и Информатика.</p>
      <p>В свободное время я занимаюсь танцами
        и рисованием.</p>
      <p>Ваши сообщения можете отправлять на мой
         e-mail: <a href="mailto:lize@ukr.net">
                  lize@ukr.net</a></p>
    </div>
  </body>
</html>
```

При редактировании Web-страниц обратите внимание, что основными ошибками являются пропущенные «парные» символы, такие как знаки начала «<» и окончания тега «>», одинарные «'» и двойные «"» кавычки и т. д.

6. Просмотрите в браузере файл `liza_0.html`.

Он должен выглядеть так:



Если полученный результат отличается от ожидаемого – повторите пункты 4 ÷ 6 еще раз.

7. На основании файла `liza_0.html` создайте Web-страницу `liza_1.html` и отредактируйте ее, установив при помощи встроенных стилей такие параметры форматирования:

а) абзац «Мне 18 лет.»: **полужирный *курсив***;

б) гиперссылка на место учебы – **ХНАГХ**: без подчеркивания;

в) название факультета – «Экономики …»: цвет – **синий**;

г) название специальности – «Учет …»: начертание – **полужирное**;

д) название любимых дисциплин: подчеркнуть;

е) название увлечений (хобби): выделить цветом, первое – **фуксии** (**fuchsia**, `#FF00FF`), второе – **бирюзовым** (**aqua**, `#00FFFF`);

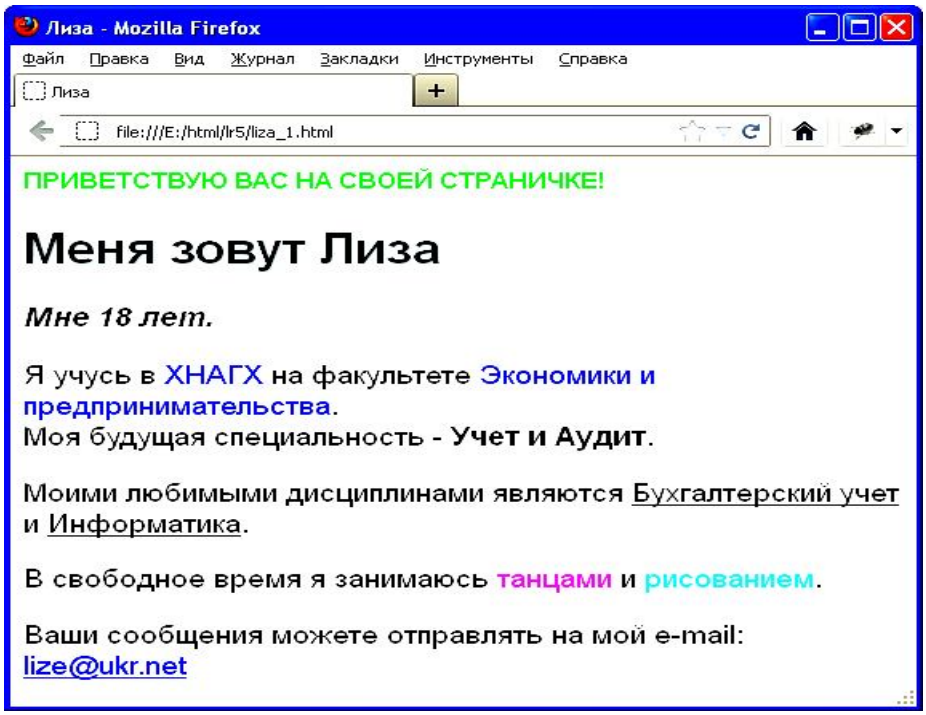
В результате она должна иметь такое содержание:

```
<html>
  <head>
    <title>Лиза</title>
  </head>
  <body style="font-family:Arial, sans-serif;
    color:dimgray;">
    <p style="text-transform:uppercase;
      color:#00FF00; font-weight:bold;">
      Приветствую Вас на своей страничке!</p>
    <h1 style="text-shadow:1px 1px #CDD9DB;">
      Меня зовут Лиза</h1>
    <div style="font-size:14pt;">
      <p style="font-weight:bold;
        font-style:italic;">Мне 18 лет.</p>
      <p>Я учусь в <a href=http://www.kname.edu.ua/
                     title="Здесь я учусь"
                     style="text-decoration:none;">
                     ХНАГХ</a>
        на факультете
        <span style="color:#0000FF;">Экономики
        и предпринимательства</span>.<br>
        Моя будущая специальность -
        <span style="font-weight:bold;">
          Учет и Аудит</span>.</p>
      <p>Моими любимыми дисциплинами являются
        <span style="text-decoration:underline;">
          Бухгалтерский учет</span> и
        <span style="text-decoration:underline;">
          Информатика</span>.</p>
      <p>В свободное время я занимаюсь
        <span style="color:fuchsia;">танцами</span> и
        <span style="color:aqua;">
          рисованием</span>.</p>
      <p>Ваши сообщения можете отправлять на мой
        e-mail: <a href="mailto:lize@ukr.net">
                  lize@ukr.net</a></p>
    </div>
  </body>
</html>
```

Обратите внимание на выделение блочных элементов при помощи тега , а строчных – тега .

8. Просмотрите в браузере отредактированный файл liza_1.html.

Он должен выглядеть так:



Если полученный результат отличается от ожидаемого – повторите два последних пункта еще раз.

Шаг 2. Внедренные стили (Embedding Styles)

1. На основании файла liza_1.html создайте Web-страницу liza_2.html и отредактируйте ее (методом Вырезать-Вставить), установив при помощи внедренных стилей такие же параметры форматирования, как и у файла liza_1.html, т. е.:

а) текст на всей странице: Arial, если такой отсутствует – любой рубленный (sans-serif), цвет – тускло-серый (dimgray – #696969);

б) абзац «Приветствую Вас …»: символы – ВСЕ ПРОПИСНЫЕ, цвет – **зеленый**, начертание – **полужирное**;

в) заголовок_1 «Меня зовут ..»: с тенью размером 1 пиксель, цвет – **#CDD9DB;**

г) весь остальной текст на странице: размер – 14 пунктов.

д) абзац «Мне 18 лет.»: **полужирный *курсив***;

е) гиперссылка на место учебы – **ХНАГХ**: без подчеркивания, оно должно появляться только при наведении указателя мыши;

ж) название факультета – «Экономики …»: цвет – **синий**;

з) название специальности – «Учет …»: начертание – **полужирное**;

и) название любимых дисциплин: подчеркнуть;

к) название увлечений (хобби): выделить цветом, первое – **фуксии** (**fuchsia**, #FF00FF), второе – **бирюзовым** (**aqua**, #00FFFF);

В результате она должна иметь такое содержание:

```
<html>
  <head>
    <title>Лиза</title>
    <style type="text/css">
      body{
        font-family:Arial, sans-serif;
        color:dimgray;
      }
      .grtng{
        text-transform:uppercase;
        color:#00FF00;
        font-weight:bold;
      }
      h1{
        text-shadow:1px 1px #CDD9DB;
      }
      div{
        font-size:14pt;
      }
      #ag{
        font-weight:bold;
        font-style:italic;
      }
      #stdy:link, #stdy:visited, #stdy:active{
        text-decoration:none;
      }
      #stdy:hover{
        text-decoration:underline;
      }
      #fclt{
        color:#0000FF;
      }
      #spclt{
        font-weight:bold;
      }
      #dscpln{
        text-decoration:underline;
      }
      #hbb1{
        color:fuchsia;
      }
      #hbb2{
        color:aqua;
      }
    </style>
  </head>
  <body>
    <p class=grtng>
      Приветствую Вас на своей страничке!</p>
    <h1>Меня зовут Лиза</h1>
    <div>
      <p id=ag>Мне 18 лет.</p>
      <p>Я учусь в <a href=http://www.kname.edu.ua/
```

```
                    title="Здесь я учусь" id=stdy>
                    ХНАГХ</a> на факультете
        <span id=fclt>Экономики
          и предпринимательства</span>.<br>
        Моя будущая специальность -
        <span id=spclt>Учет и Аудит</span>.</p>
      <p>Моими любимыми дисциплинами являются
        <span id=dscpln>Бухгалтерский учет</span> и
        <span id=dscpln>Информатика</span>.</p>
      <p>В свободное время я занимаюсь
        <span id=hbb1>танцами</span> и
        <span id=hbb2>рисованием</span>.</p>
      <p>Ваши сообщения можете отправлять на мой
        e-mail: <a href="mailto:lize@ukr.net">
                lize@ukr.net</a></p>
    </div>
  </body>
</html>
```

2. Просмотрите в браузере отредактированный файл `liza_2.html`.

Он должен выглядеть так, как и представленный выше файл `liza_1.html`.

Шаг 3. Связанные или внешние стили (Linking Styles or External Styles)

1. Создайте пустой текстовый файл `prsnl.css`.
2. Сохраните файл `liza_2.html` под именем `liza.html`.
3. В текстовом редакторе «Вырежьте» из файла `liza.html` внедренную таблицу стилей – участок кода, который находится между тегами `<style type="text/css">…</style>` (без самих тегов) и «Вставьте» в созданный ранее пустой файл `prsnl.css`.
4. Отредактируйте файл `prsnl.css`, убрав лишние пробелы в начале каждой строки.

В результате он должна иметь такое содержание:

```
body{
  font-family:Arial, sans-serif;
  color:dimgray;
}
.grtng{
  text-transform:uppercase;
  color:#00FF00;
  font-weight:bold;
}
h1{
  text-shadow:1px 1px #CDD9DB;
}
div{
  font-size:14pt;
}
#ag{
```

```
  font-weight:bold;
  font-style:italic;
}
#stdy:link, #stdy:visited, #stdy:active{
  text-decoration:none;
}
#stdy:hover{
  text-decoration:underline;
}
#fclt{
  color:#0000FF;
}
#spclt{
  font-weight:bold;
}
#dscpln{
  text-decoration:underline;
}
#hbb1{
  color:fuchsia;
}
#hbb2{
  color:aqua;
}
```

5. Сохраните отредактированный файл `prsnl.css`.

6. В файле `liza.html` замените оставшиеся там теги определения внедренной таблицы стилей `<style type="text/css">…</style>` на тег ссылки на внешнюю (связанную) таблицу стилей `prsnl.css` – `<link rel="stylesheet" type="text/css" href="prsnl.css">`.

В результате он должна иметь такое содержание:

```
<html>
  <head>
    <title>Лиза</title>
    <link rel="stylesheet" type="text/css"
      href="prsnl.css">
  </head>
  <body>
    <p class=grtng>
      Приветствую Вас на своей страничке!</p>
    <h1>Меня зовут Лиза</h1>
    <div>
      <p id=ag>Мне 18 лет.</p>
      <p>Я учусь в <a href=http://www.kname.edu.ua/
                    title="Здесь я учусь" id=stdy>
                    ХНАГХ</a> на факультете
        <span id=fclt>Экономики
          и предпринимательства</span>.<br>
        Моя будущая специальность -
        <span id=spclt>Учет и Аудит</span>.</p>
      <p>Моими любимыми дисциплинами являются
        <span id=dscpln>Бухгалтерский учет</span> и
```

```
        <span id=dscpln>Информатика</span>.</p>
      <p>В свободное время я занимаюсь
        <span id=hbb1>танцами</span> и
        <span id=hbb2>рисованием</span>.</p>
      <p>Ваши сообщения можете отправлять на мой
        e-mail: <a href="mailto:lize@ukr.net">
                lize@ukr.net</a></p>
    </div>
  </body>
</html>
```

7. Сохраните отредактированный файл `liza.html`.

8. Посмотрите в браузере полученный результат – Web-страницу `liza.html`.

Она должна выглядеть так же, как и файлы `liza_1.html` и `liza_2.html`.

9. На основе Web-страницы `liza.html` создайте аналогичные HTML-документы для следующих 4-х женских имен: Ира, Катя, Света и Наташа, соответственно, `ira.html`, `katya.html`, `sveta.html` и `natasha.html`, подключив к ним ту же таблицу стилей – `prsnl.css`.

10. Посмотрите созданные Web-страницы в браузере.

Они должны выглядеть так же, как созданные ранее Web-страницы `liza_1.html`, `liza_2.html` и `liza.html`, так как у них используется один и тот же стилевой файл `prsnl.css`.

Шаг 4. Контейнерный Web-дизайн

1. При помощи графического редактора создайте логотип размером 80×80 точек (пикселей), например, кружок зеленого цвета, и сохраните его в формате `.gif` в файле под именем `logo.gif`.

2. Создайте содержание (контент) стартовой страницы миниWeb-сайта – файл `index.html` так, чтобы он имел следующие основные фрагменты:

а) заголовок Web-сайта, состоящий из: логотипа – файл `logo.gif`. и названия – «Неразлучные подружки»,

б) полоса навигации (меню) и

в) основное содержимое – подгружаемые при помощи меню и плавающего фрейма (тег `<iframe>`) личные страницы Лизы, Иры, Кати, Светы и Наташи.

Основные составляющие миниWeb-сайта выделите при помощи тегов `<div>...</div>`, пометив их (при помощи атрибутов `id=`), соответственно: `head` – для заголовка и `menu` – для навигации. Тег `<iframe>`, с помощью которого подгружаются личные страницы, помечать не нужно – на странице он единственный.

В результате файл `index.html` должен иметь такое содержание:

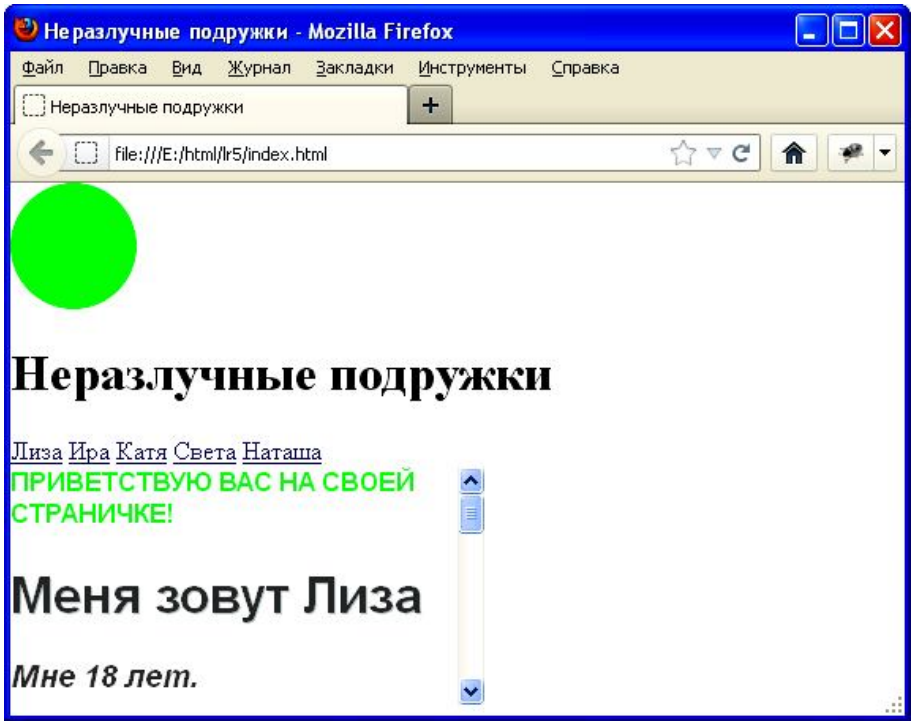
```
<html>
  <head>
    <title>Неразлучные подружки</title>
  </head>
  <body topmargin="0" leftmargin="0"
    rightmargin="0" bottommargin="0">
    <div id="head">
      <img src="logo.gif">
```

```
      <h1>Неразлучные подружки</h1>
    </div>
    <div id="menu">
      <a href="liza.html" target="content">Лиза</a>
      <a href="ira.html" target="content">Ира</a>
      <a href="katya.html" target="content">Катя</a>
      <a href="sveta.html" target="content">Света</a>
      <a href="natasha.html"
        target="content">Наташа</a>
    </div>
    <iframe src="liza.html" name="content"
      frameborder="0" marginwidth="0"
      marginheight="0">Чтобы увидеть эту
      страницу обновите браузер</iframe>
  </body>
</html>
```

Атрибуты …margin…= в тегах <body> и <iframe> установлены в нулевые значения для того, чтобы убрать внутренние отступы, которые по умолчанию оставляют некоторые браузеры, с целью более точного позиционирования в них элементов.

3. Просмотрите в браузере созданный миниWeb-сайт.

Он должен выглядеть так:



Проверьте работу миниWeb-сайта. По умолчанию браузеры создают плавающий фрейм (куда мы подгружаем персональную информацию) размером 300×150 пикселей.

4. Создайте такую разметку для миниWeb-сайта:

Логотип	Название
Меню	
Содержание	

В верхнем поле должен отображаться заголовок Web-сайта с логотипом и названием. Ниже располагается меню, при помощи которого в оставшееся пространство будут загружаться персональные страницы.

Необходимо также обеспечить:

а) отступ вокруг заголовка 10 пикселей, чтобы он не «прилипал» к границам окна;

б) логотип выровнять по левому краю заголовка;

в) название сайта: выровнять по правому краю заголовка, символы – ВСЕ ПРОПИСНЫЕ, шрифт – Arial или любой рубленый, цвет – `#FF00CC`, дополнительное расстояние между символами (разгонка) – 1 мм;

г) меню: расположить – по центру, отступ внизу – 10 пикселей, ниже – полоска шириной 5 пикселей, сплошная желтого цвета, чтобы отделить меню от основного содержания;

д) пункты меню: в виде выпуклых кнопок, цвет – `#3355cc`, надписи: цвет – белый, шрифт: **полужирный**, среднего размера, tahoma, без подчеркивания;

е) основное содержание (тег `<iframe>`): отступ вокруг содержания – 0.5 ширины символа, чтобы содержание не «прилипало» к краям, высота – 82 % от высоты окна браузера, ширина – 98 % от ширины окна браузера, чтобы полностью использовать оставшуюся площадь окна для отображения страниц с персональной информацией.

Для этого создайте внешнюю таблицу стилей – файл `position_1.css` – такого содержания:

```
#head{
  padding:10px; /* Отступ вокруг заголовка */
}
#head img{ /* Логотип */
  float:left; /* Влево */
}
#head h1{ /* Заголовок */
  text-align:right; /* Вправо */
  text-transform:uppercase; /* ВСЕ ПРОПИСНЫЕ */
  font-family:Arial, sans-serif; /* Рубленный */
  color:#FF00CC;
  letter-spacing:1mm; /* Разгонка 1мм */
}
#menu{
  text-align:center; /* По центру */
  padding-bottom:10px; /* Отступ ниже меню */
  border-bottom:5px solid yellow; /* Отделить */
}
#menu a{
  padding:2px 10px; /* Отступы вокруг названий */
  border:solid 1px; /* Выпуклые кнопки */
  border-color:#4466ff #223388 #223388 #4466ff;
  background:#3355cc;
  color:white; /* Параметры надписей */
  font:bold medium tahoma;
  text-decoration:none; /* Без подчеркивания */
}
```

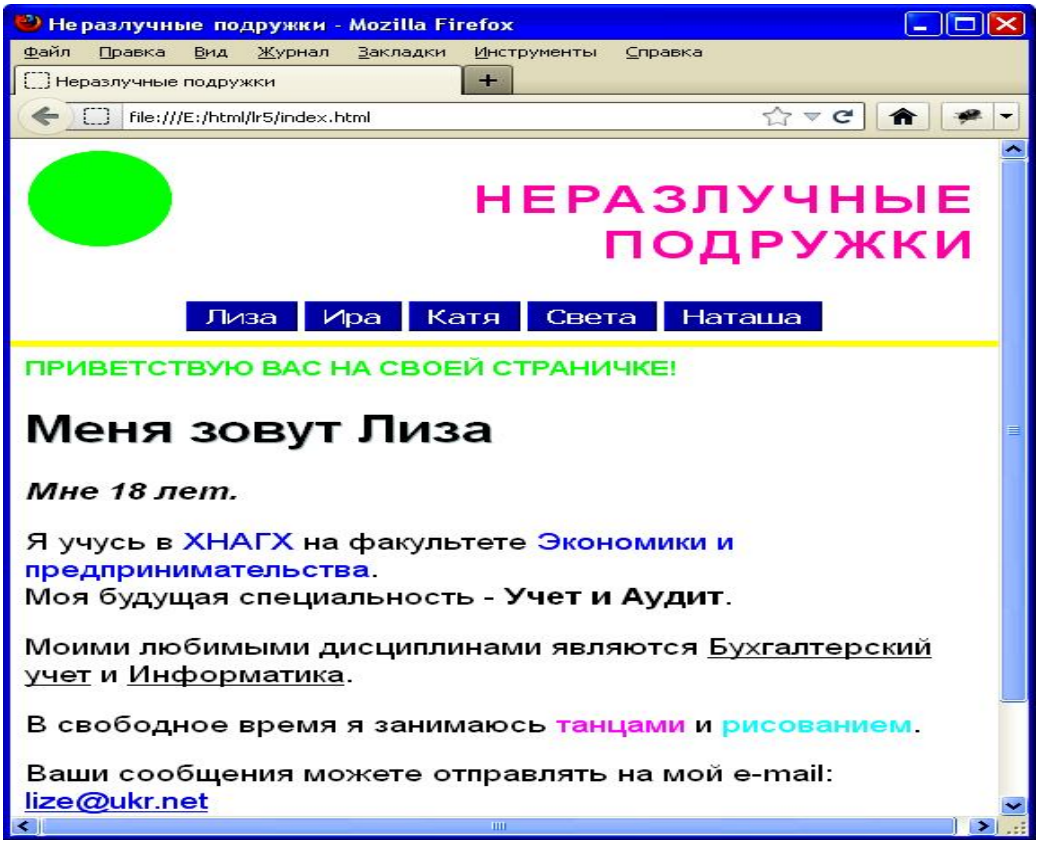
```
iframe{ /* Содержание */
  padding:0.5em; /* Отступ вокруг 0.5 символа */
  height:82%;
  width:98%;
}
```

5. Подключите созданную таблицу стилей к миниWeb-сайту – файлу `index.html`.

Для этого добавьте в раздел `head` ссылку на таблицу стилей – тег `<link rel="stylesheet" type="text/css" href="position_1.css">`.

6. Просмотрите в браузере отредактированный файл `index.html`.

Он должен выглядеть так:



7. Создайте следующую разметку миниWeb-сайта:

Логотип	Название
Меню	Содержание

В верхнем поле должен отображаться заголовок Web-сайта с логотипом и названием. Ниже слева располагается меню, при помощи которого в оставшееся, правое, пространство будут загружаться персональные данные.

Необходимо также обеспечить:

а) отступ вокруг заголовка 10 пикселей, чтобы он не «прилипал» к границам окна, внизу – полоска шириной 5 пикселей, сплошная **желтого** цвета, чтобы отделить заголовок от основного содержания;

б) логотип выровнять по левому краю заголовка;

в) название сайта: выровнять по правому краю заголовка, символы – ВСЕ ПРОПИСНЫЕ, шрифт – Arial или любой рубленый, цвет – **#FF00CC**, дополнительное расстояние между символами (разгонка) – 1 мм;

г) меню: ниже заголовка, прижать к левой стороне окна, ширина – 90 пикселей, чтобы помешались названия пунктов, на всю оставшуюся высоту окна, цвет фона – **#4169e1**, отступ сверху – 1 символ, чтобы пункты не «прилипали» к верху;

д) пункты меню: расположить – сверху вниз, отступ слева – 0.5 символа, надписи: цвет – **#ffd700**, шрифт: **полужирный**, среднего размера, arial или любой рубленный, без подчеркивания;

е) выбранный пункт меню: выделить цветом, символы – **#333399**, фон – **#ccfcfc**;

ж) основное содержание (тег <iframe>): отступ вокруг содержания – 0.5 ширины символа, чтобы содержание не «прилипало» к краям, позиционирование – абсолютное, левый край – справа от меню, т. е. начать с 90-го пикселя, высота – 85 % от высоты окна браузера, ширина – 92 % от ширины окна браузера, чтобы полностью использовать оставшуюся площадь окна для отображения страниц с персональной информацией.

Для этого на основании предыдущей таблицы стилей – файла position_1.css – создайте новую внешнюю таблицу стилей – файл position_2.css – такого содержания:

```
#head{
  padding:10px; /* Отступ вокруг заголовка */
  border-bottom:5px solid yellow; /* Отделить */
}
#head img{ /* Логотип */
  float:left; /* Влево */
}
#head h1{ /* Заголовок */
  text-align:right; /* Вправо */
  text-transform:uppercase; /* ВСЕ ПРОПИСНЫЕ */
  font-family:Arial, sans-serif; /* Рубленный */
  color:#FF00CC;
  letter-spacing:1mm; /* Разгонка 1мм */
}
#menu{ /* Меню */
  float:left; /* Влево */
  width:90px;
  height:87%; /* До самого низа окна */
  background:#4169e1;
  padding-top:1ex; /* Отступ сверху */
}
#menu a{ /* Пункты меню */
  display:block; /* Сверху-вниз */
  padding-left:0.5em; /* Отступ слева 0.5 символа */
  color:#ffd700; /* Параметры надписей */
  font:bold medium arial, san-serif;
  text-decoration:none; /* Без подчеркивания */
```

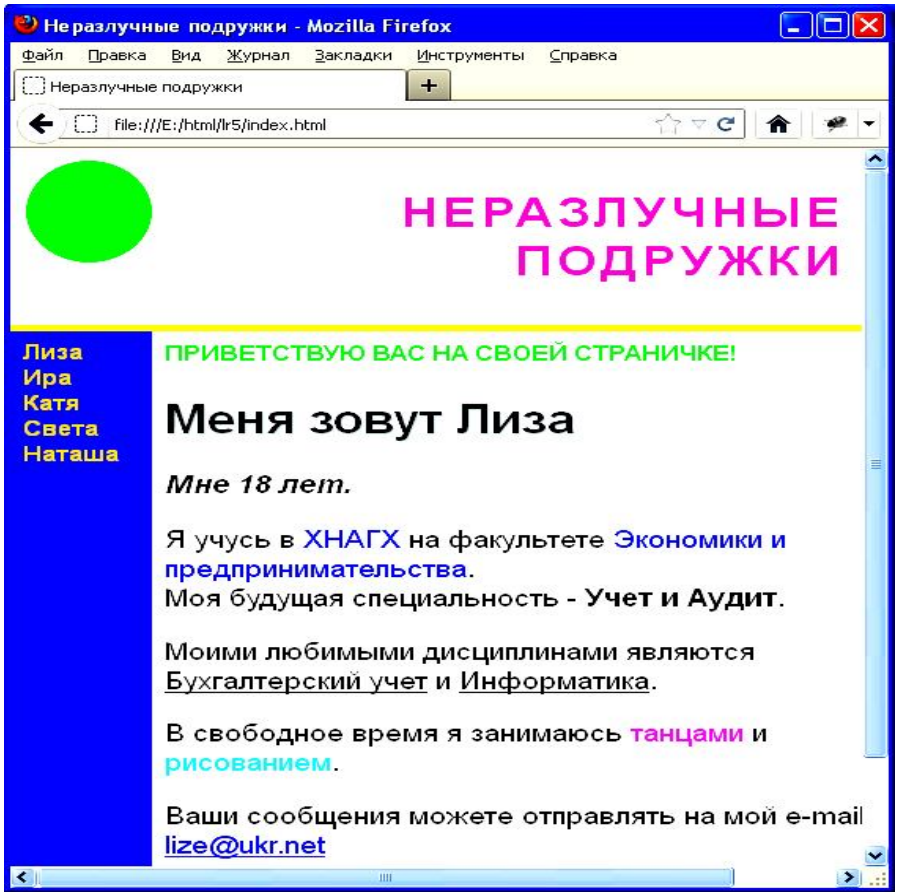
```
}
#menu a:hover{ /* Текущий пункт меню - выделить */
  color:#333399;
  background:#ccfcfc;
}
iframe{ /* Содержание */
  padding:0.5em; /* Отступ вокруг 0.5 символа */
  position:absolute;
  left:90px; /* Левее меню */
  height:85%;
  width:92%;
}
```

8. Подключите к миниWeb-сайту – файлу `index.html` – новую таблицу стилей – файл `position_2.css`.

Для этого в разделе `head` в теге `<link>` замените ссылку (значение атрибута `href=`) с `position_1.css` на `position_2.css`.

9. Просмотрите в браузере отредактированную разметку миниWeb-сайта.

Сайт должен выглядеть так:



10. Создайте еще и такую разметку миниWeb-сайта:

<table>
<tr><td rowspan="2">Меню</td><td>Логотип Название</td></tr>
<tr><td>Содержание</td></tr>
</table>

В самом левом поле располагается меню, при помощи которого в правое нижнее поле будут загружаться персональные страницы. В правом верхнем поле должен отображаться заголовок Web-сайта с логотипом и его названием.

Необходимо также обеспечить:

а) заголовок: отступ вокруг – 10 пикселей, чтобы он не «прилипал» к границам окна, внизу – полоска шириной 5 пикселей, сплошная **желтого** цвета, чтобы отделить заголовок от основного содержания, позиционирование – относительное, сместить вправо на 90 пикселей, чтобы оставить место для меню;

б) логотип выровнять по левому краю заголовка;

в) название сайта: выровнять по правому краю заголовка, символы – ВСЕ ПРОПИСНЫЕ, шрифт – Arial или любой рубленый, цвет – **#FF00CC**, дополнительное расстояние между символами (разгонка) – 1 мм;

г) меню: в самой левой стороне окна, ширина – 90 пикселей, чтобы помешались названия пунктов, на всю высоту окна, цвет фона – **#4169e1**, отступ сверху – 1 символ, чтобы пункты не «прилипали» к верху;

д) пункты меню: расположить – сверху вниз, отступ слева – 0.5 символа, надписи: цвет – **#ffd700**, шрифт: **полужирный**, среднего размера, arial или любой рубленный, без подчеркивания;

е) выбранный пункт меню: выделить цветом, символы – **#333399**, фон – **#ccfcfc**;

ж) основное содержание (тег `<iframe>`): отступ вокруг содержания – 0.5 ширины символа, чтобы содержание не «прилипало» к краям, позиционирование – абсолютное, левый край – справа от меню, т. е. начать с 90-го пикселя, высота – 85 % от высоты окна браузера, ширина – 92 % от ширины окна браузера, чтобы полностью использовать оставшуюся площадь окна для отображения страниц с персональной информацией.

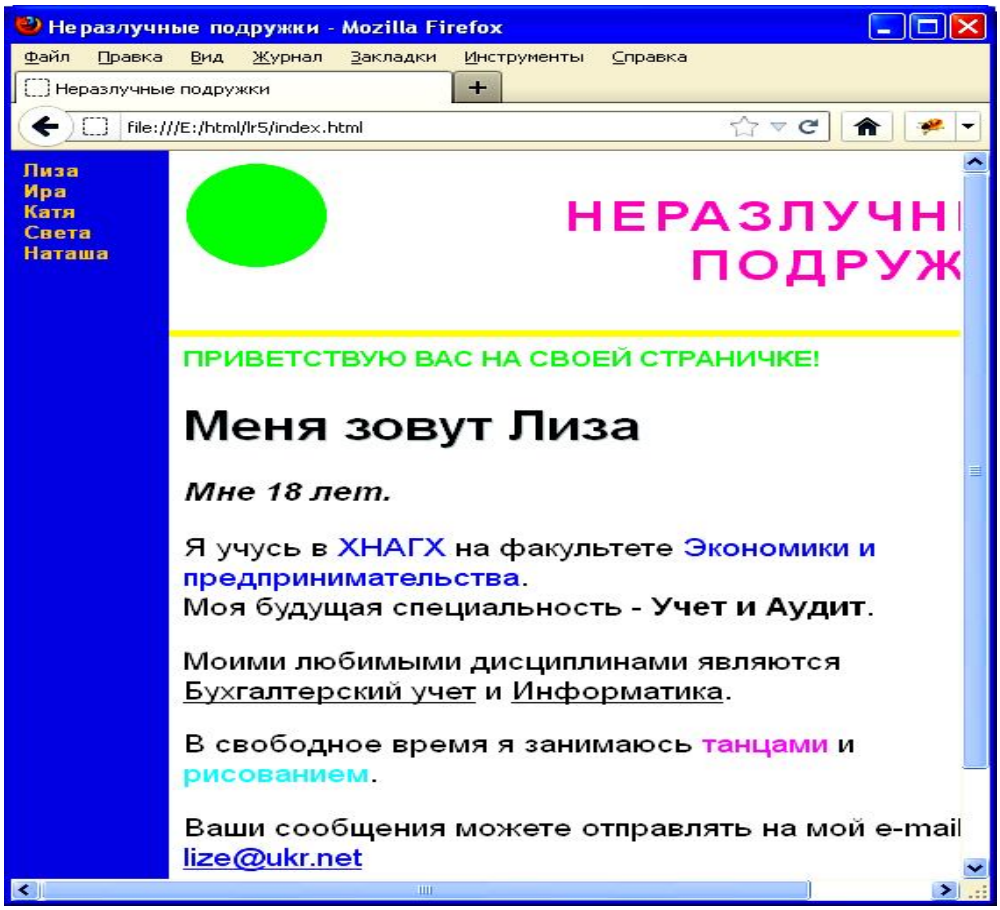
Для этого на основании предыдущей таблицы стилей – файла `position_2.css` – создайте новую внешнюю таблицу стилей – файл `position_3.css` – такого содержания:

```
#head{
  padding:10px; /* Отступ вокруг заголовка */
  border-bottom:5px solid yellow; /* Отделить */
  position:relative; /* Относительно исходного */
  left:90px; /* Оставить место для меню */
  width:92%; /* На оставшуюся ширину окна */
}
#head img{ /* Логотип */
  float:left; /* Влево */
}
#head h1{ /* Заголовок */
  text-align:right; /* Вправо */
  text-transform:uppercase; /* ВСЕ ПРОПИСНЫЕ */
  font-family:Arial, sans-serif; /* Рубленный */
  color:#FF00CC;
  letter-spacing:1mm; /* Разгонка 1мм */
}
#menu{ /* Меню */
  position:absolute;
  top:0px; /* С самого верха окна */
  width:90px;
```

```
  height:99%; /* До самого низа окна */
  background:#4169e1;
  padding-top:1ex; /* Отступ сверху */
}
#menu a{ /* Пункты меню */
  display:block; /* Сверху-вниз */
  padding-left:0.5em; /* Отступ слева 0.5 символа */
  color:#ffd700; /* Параметры надписей */
  font:bold small arial, san-serif;
  text-decoration:none; /* Без подчеркивания */
}
#menu a:hover{ /* Текущий пункт меню - выделить */
  color:#333399;
  background:#ccfcfc;
}
iframe{ /* Содержание */
  padding:0.5em; /* Отступ вокруг 0.5 символа */
  position:absolute;
  left:90px; /* Левее меню */
  height:85%;
  width:92%;
}
```

11.Подключите к миниWeb-сайту – файлу `index.html` – вновь созданную таблицу стилей – файл `position_3.css`.

12.Просмотрите в браузере отредактированную разметку миниWeb-сайта. Он должен выглядеть так:



13.Самостоятельно измените значения свойств позиционирования, или размеров, элементов миниWeb-сайта и посмотрите как «поплывет» разметка, особенно, при изменении размеров окна браузера, а так же – в разных браузерах.

Шаг 5. Бонус

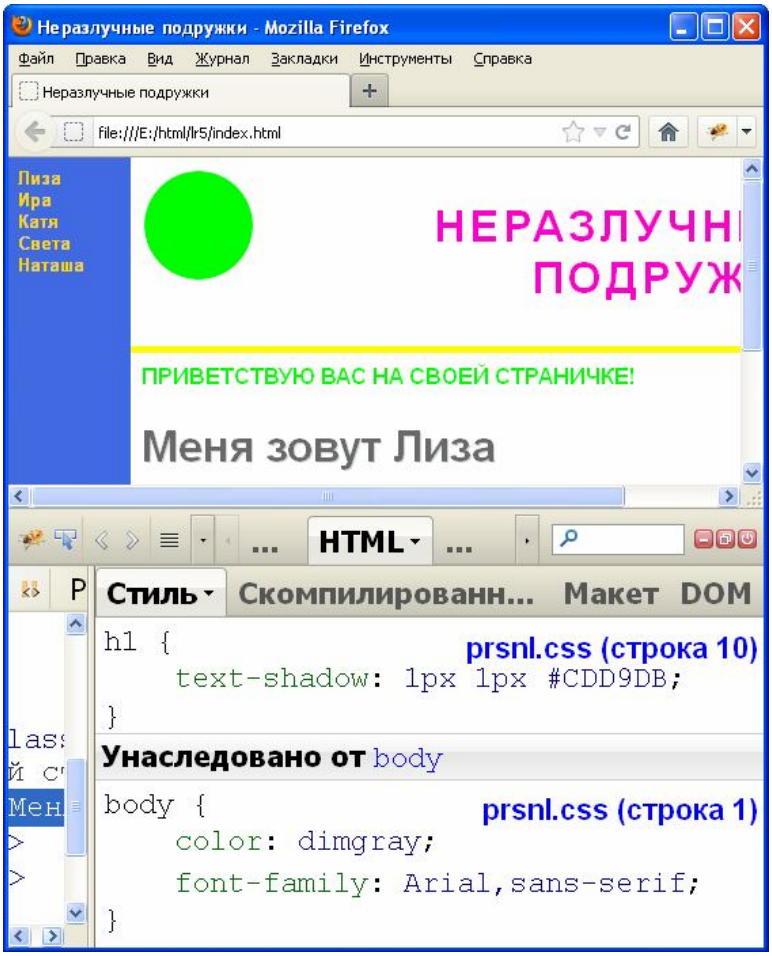
Для того, чтобы воспользоваться бонусом, у Вас должен быть:

1. установлен браузер Firefox и

2. в браузере Firefox установлено бесплатное дополнение (расширение, плагин, англ. plug-in) Firebug.

1. Щелкните правой кнопкой мыши по любому элементу Web-страницы, например, по тексту «**Меня зовут …**» на сайте **НЕРАЗЛУЧНЫЕ ПОДРУЖКИ** и в раскрывшемся контекстном меню выберите команду **Инспектировать элемент с помощью Firebug**.

В результате по умолчанию в нижней части окна браузера раскроется дополнительное окно, которое состоит из 2-х частей. В правой части этого окна отображается HTML-код выделенного элемента, а в левой – соответствующие ему CSS-правила.



2. Перемещаете указатель мыши по HTML-коду в левой части нижнего окна и наблюдайте, как в основном окне браузера будут выделяться соответствующие элементы Web-страницы.

3. В окне Firebug щелкните по кнопке  **Щелкните на элементе страницы для анализа**.

В результате при перемещении указателя мыши над любым элементом в основном окне браузера он будет выделяться, а соответствующий ему HTML- и CSS-код будет отображаться в окне Firebug.

4. В окне Firebug можно «на лету» редактировать HTML- и CSS-код.

При этом страница в браузере немедленно обновляется, отображая сделанные изменения. Однако …

⚠ *Сделанные изменения кода в Firebug прямо в файле сохранить нельзя. Для сохранения сделанных изменений их необходимо в окне Firebug выделить и скопировать в буфер обмена, а затем, вставить во внешнем редакторе.*

Больше информации о Firebug на русском языке можно посмотреть на сайте firebug.ru. Оригинальный же сайт Firebug – http://getfirebug.com.

Аналогичные средства исследования и отладки кода имеются в большинстве современных браузеров. Так, например, в Opera оно называется Dragonfly, в Safari – WebInspector, а в Chrome и Internet Explorer – Developer Tools.

Индивидуальные задания

1. Создайте каталог `iz` и сделайте его текущим.
2. Создайте личный миниWeb-сайт, на котором будете отображать информацию о:
   - себе,
   - родителях,
   - друзьях,
   - любимых предметах,
   - увлечениях,
   - и т. д.

Разметка главной страница должна быть выполнена при помощи контейнерного Web-дизайна. Она должна содержать минимум 4 области: логотип, название, меню и содержание. За основу можете взять одну из выполненных и понравившейся разметок.

3. Обеспечьте одинаковое отображение сайта в различных браузерах.

Начинать разработку сайта лучше в браузере Firefox, а затем уже добиваться одинакового представления во всех остальных.

4. В тетрадь для выполнения лабораторных работ занесите эскизы созданных Web-страниц и, соответствующие им, тексты HTML-документов и CSS-таблиц.

Контрольные вопросы

1. Что такое каскадные таблицы стилей? Зачем они нужны? Как используются?
2. Что такое стиль? Что такое таблица стилей?
3. Что такое каскадирование? Приведите пример.
4. Что такое наследование? Приведите пример.

5. Что такое правило CSS? Объясните его состав и структуру.
6. Что такое селектор, декларация, свойство и значение?
7. Зачем нужен селектор? Перечислите основные типы селекторов.
8. Что такое селектор по элементу? Приведите пример.
9. Что такое селектор по классу? Приведите пример.
10. Что такое селектор по идентификатору? Приведите пример.
11. Что такое контекстный селектор? Приведите пример.
12. Что такое псевдоселекторы? Зачем они нужны? Как используются? Приведите пример.
13. Что такое группировка? Какие они бывают? Зачем они нужны? Приведите примеры.
14. Какие классы единиц измерения предусмотрены в CSS? Перечислите основные из этих классов.
15. Перечислите способы применения CSS к HTML-документу?
16. Приведите пример встроенного стиля (Inline Style).
17. Приведите пример внедренного стиля (Embedding Style).
18. Приведите пример внешнего или связанного стиля (Linking Style или External Style).
19. Каков порядок расстановки приоритетов применения стилей из различных источников к элементу Web-страницы?
20. Как можно «нарушить» установленные правила каскадирования?
21. Что такое блочный элемент Web-страницы? Приведите примеры.
22. Что такое строчный элемент Web-страницы? Приведите примеры.
23. Как превратить блочный элемент Web-страницы в строчный, и – наоборот? Приведите пример.
24. Объясните назначение тега `<div>`.
25. Объясните назначение тега `<span>`.
26. Что общего, и в чем разница между тегами `<div>` и `<span>`?
27. Объясните назначение свойства `position:`. Какие значения оно может принимать?
28. Как в CSS реализуются плавающие блоки? Какие значения может принимать это свойство?
29. При помощи какого свойства CSS можно всегда расположить текущий блок ниже любого плавающего? Какие значения оно может принимать?
30. Как создать тень у текста? Можно ли добиться такого же эффекта средствами HTML?
31. Сформулируйте и задайте свой вопрос по пройденному материалу.
32. Сформулируйте и внесите свое предложение по совершенствованию темы.

Список источников

1. Далхаймер К., Уэлш М. Запускаем Linux, 5-е издание. – Пер. с англ. – СПб.: Символ Плюс, 2008. – 992 с.: ил.

2. Колисниченко Д.Н. Linux. От новичка к профессионалу. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2010. – 784 с.: ил.

3. Колисниченко Д.Н. Самоучитель Linux. Установка, настройка, использование. – 4-е изд., перераб. и доп. – СПб.: Наука и Техника, 2006. – 688 с: ил.

4. Колисниченко Д.Н. Самоучитель системного администратора Linux. – СПб.: БХВ-Петербург, 2011. – 544 с.: ил.

5. Костромин В.А. Самоучитель Linux для пользователя. – СПб.: БХВ-Петербург, 2005. – 672 с.: ил.

6. Дронов В.А. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов. – СПб.: БХВ-Петербург, 2011. – 416 с.: ил.

7. Коржинский С.Н. Настольная книга Web-мастера: эффективное применение HTML, CSS, JavaScript. Издание второе, исправленное и дополненное. – М.: Издательский торговый дом «КноРус», 2000. – 320 с.

8. Матросов А.А., Сергеев А.О., Чаунин М.П. HTML 4.0. – БХВ-Петербург, 2004. – 672 с.: ил.

Навчальне видання

Методичні вказівки
для виконання лабораторних, практичних,
самостійних і контрольних робіт
з курсу

«**ТЕХНІЧНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ»**

(для студентів 5-го курсу денної та 6-го курсу заочної форм навчання магістрів спеціалності 8.18010013 «Управління проектами»)

(рос. мовою)

Укладачі: **ПОГРЕБНЯК** Борис Іванович
ВИСОЦЬКА Галина Василівна

Відповідальний за випуск: *О. Б. Костенко*

За авторською редакцією

Комп'ютерне верстання *К. А. Алексанян*

План 2013, поз. 374М

Підп. до друку 17.05.2013р.	Формат 60х84/16
Друк на ризографі	Ум. друк. арк. 7,1
Тираж 50 пр.	Зам. №

Видавець і виготовлювач:
Харківський національний університет
міського господарства імені О. М. Бекетова
вул. Революції, 12, Харків, 61002
Електронна адреса: rectorat@ksame.kharkov.ua
Свідоцтво суб'єкта видавничої справи:
ДК №4064 від 12.05.2011